

# About Component Integration Laboratories

OpenDoc is presented and maintained through a nonprofit organization devoted to promoting cross-platform standards, architectures, and protocols in a vendor-independent fashion. This organization, Component Integration Laboratories (CI Labs), is composed of a number of platform and application vendors with a common interest in solving OpenDoc issues and promoting interoperability.

CI Labs supports several levels of participation through different membership categories. If you are interested in shaping the future direction of component software, or if you simply need to be kept abreast of the latest developments, you can become a member. For an information packet, send your mailing address to:

Component Integration Laboratories  
PO Box 61747  
Sunnyvale, CA 94088-1747

Telephone:	408-864-0300
FAX:	408-864-0380
Internet:	<a href="mailto:cilabs@cilabs.org">cilabs@cilabs.org</a>
World Wide Web:	<a href="http://www.cilabs.org">http://www.cilabs.org</a>

---

## How to Use the OpenDoc Programming Reference

This reference is a detailed technical guide and reference for application programmers creating programs using the OpenDoc interface for OS/2 Warp. It gives reference information and code examples to enable you to write source code using Workplace classes and methods.

Before you begin to use this information, it would be helpful to understand how you can:

- Expand the Contents to see all available topics
- Obtain additional information for a highlighted word or phrase
- Use action bar choices
- Use the programming information.

### How to Use the Contents

When the Contents window first appears, some topics have a plus (+) sign beside them. The plus sign indicates that additional topics are available.

To expand the Contents if you are using a mouse, click on the plus sign. If you are using the keyboard, use the Up or Down Arrow key to highlight the topic, and press the plus (+) key. For example, **Code Pages** has a plus sign beside it. To see additional topics for that heading, click on the plus sign or highlight that topic and press the plus (+) key.

To view a topic, double-click on the topic (or press the Up or Down Arrow key to highlight the topic, and then press the Enter key).

### How to Obtain Additional Information

After you select a topic, the information for that topic appears in a window. Highlighted words or phrases indicate that additional information is available. You will notice that certain words and phrases are highlighted in green letters, or in white letters on a black background. These are called hypertext terms. If you are using a mouse, double-click on the highlighted word. If you are using a keyboard, press the Tab key to move to the highlighted word, and then press the Enter key. Additional information then appears in a window.

### How to Use Action Bar Choices

Several choices are available for managing information presented in the *OpenDoc Programming Reference*. There are three pull-down menus on the action bar: the **Services** menu, the **Options** menu, and the **Help** menu.

The actions that are selectable from the **Services** menu operate on the active window currently displayed on the screen. These actions include the following:

#### Bookmark

Allows you to set a placeholder so you can retrieve information of interest to you.

When you place a bookmark on a topic, it is added to a list of bookmarks you have previously set. You can view the list, and you can remove one or all bookmarks from the list. If you have not set any bookmarks, the list is empty.

To set a bookmark, do the following:

1. Select a topic from the Contents.
2. When that topic appears, choose the **Bookmark** option from the **Services** pull-down.

3. If you want to change the name used for the bookmark, type the new name in the field.
4. Click on the **Place** radio button (or press the Up or Down Arrow key to select it).
5. Click on **OK** (or select it and press Enter). The bookmark is then added to the bookmark list.

### Search

Allows you to find occurrences of a word or phrase in the current topic, selected topics, or all topics.

You can specify a word or phrase to be searched. You can also limit the search to a set of topics by first marking the topics in the Contents list.

To search for a word or phrase in all topics, do the following:

1. Choose the **Search** option from the **Services** pull-down.
2. Type the word or words to be searched for.
3. Click on **All sections** (or press the Up or Down Arrow keys to select it).
4. Click on **Search** (or select it and press Enter) to begin the search.
5. The list of topics where the word or phrase appears is displayed.

### Print

Allows you to print one or more topics. You can also print a set of topics by first marking the topics in the Contents list.

To print the document Contents list, do the following:

1. Choose **Print** from the **Services** pull-down.
2. Click on **Contents** (or press the Up or Down Arrow key to select it).
3. Click on **Print** (or select it and press Enter).
4. The Contents list is printed on your printer.

### Copy

Allows you to copy a topic that you are viewing to the System Clipboard or to a file that you can edit. You will find this particularly useful for copying syntax definitions and program samples into the application that you are developing.

You can copy a topic that you are viewing in two ways:

- **Copy** copies the topic that you are viewing into the System Clipboard. If you are using a Presentation Manager editor (for example, the System Editor) that copies or cuts (or both) to the System Clipboard, and pastes to the System Clipboard, you can easily add the copied information to your program source module.
- **Copy to file** copies the topic that you are viewing into a temporary file named TEXT.TMP. You can later edit that file by using any editor. You will find TEXT.TMP in the directory where your viewable document resides.

To copy a topic, do the following:

1. Expand the Contents list and select a topic.
2. When the topic appears, choose **Copy to file** from the **Services** pull-down.
3. The system puts the text pertaining to that topic into the temporary file named TEXT.TMP.

For information on one of the other choices in the **Services** pull-down, highlight the choice and press the F1 key.

The actions that are selectable from the **Options** menu allow you to change the way your Contents list is displayed. To expand the Contents and show all levels for all topics, choose **Expand all** from the **Options** pull-down. You can also press the Ctrl and \* keys together. For information on one of the other choices in the **Options** pull-down, highlight the choice and press the F1 key.

The actions that are selectable from the **Help** menu allow you to select different types of help information. You can also press the F1 key for help information about the Information Presentation Facility (IPF).

### How to Use the Programming Information

This document consists of guide and reference information that provides a detailed description of each function, message, constant, and data type. It provides language-dependent information about the functions which enable the user to generate call statements in the C Language.

Workplace Shell programming information is presented by component, such as Workplace Classes, Instance Methods, and Class Methods, for example:

## Contents

- + Notices
- + How to Use the OpenDoc Class Reference
- + Introduction to OpenDoc
- + OpenDoc Classes and Methods

By clicking on the plus sign beside "OpenDoc Classes and Methods", you see an alphabetic list of the OpenDoc classes. By clicking on the plus sign beside one of the classes, you see an alphabetic list of the methods. Selecting a method takes you directly into the reference information for that method.

Units of reference information are presented in selectable multiple windows or viewports. A viewport is a Presentation Manager window that can be sized, moved, minimized, maximized, or closed. By selecting a unit (in this case, an entry on the Contents list), you will see two windows displayed:

Unit Title	Selection Title
Select an item:	
Syntax	
Returns	
Remarks	
Related Methods	
Override Policy	
Exceptions	
Glossary	

The window on the left is the primary window. It contains a list of items that are always available to you. The window on the right is the secondary window. It contains a "snapshot" of the unit information. For reference units (that is, method descriptions), this window contains the Method Syntax.

All of the information needed to understand a reference unit (or topic) is readily available to you through the primary window. The information is divided into discrete information groups, and only the appropriate information group appears for the topic that you are viewing.

The information groups for a reference unit (that is, a method description) can include all or some of the following:

- Syntax
- Returns
- Remarks
- Related Methods
- Override Policy
- Exceptions
- Glossary

This list may vary. Some topics may be omitted when they do not apply.

Information groups are displayed in separate viewports that are stacked in a third window location that overlaps the secondary window. By selecting an item (information group) in the primary window, the item is displayed in the third window location, as follows:

Unit Title	Selection	Glossary
Select an item:		Select a starting letter of glossary terms
.		
.		
.		A N
.		B O
.		C P
Glossary		.
		.
		.
		M Z

By selecting successive items from the primary window, additional windows are displayed on top of the previous windows displayed in the third window location. For example, in a function description, Parameters and Return Values are items listed in the primary window. When selected, they appear one on top of the other in the third window location. Because of this, you may move the first selected (topmost) window to the left before selecting the next item. This allows simultaneous display of two related pieces of information from the "stack" of windows in

the third window location, as follows:

Unit Title	Parameters	Return Values
Select an item		
.		
.		
.		
Returns		
Errors		
.		
.		
.		

Each window can be individually closed from its system menu. All windows are closed when you close the primary window.

Some secondary windows may have the appearance of a split screen. For example, an illustration may appear in the left half of the window, and scrollable, explanatory information may appear in the right half of the window. Because illustrations may not necessarily fit into the small window size on your screen, you may maximize the secondary window for better readability.

---

## Conventions Used in this Reference

The purpose of this reference is to give information about classes, methods, constants, and data types. It provides information about the methods which enables the user to call functions in the C programming language.

The following information is provided:

- The syntax and parameters for each method.
- The syntax of each data type and structure

---

## OpenDoc Fundamentals

OpenDoc is an architecture designed to enable the construction of compound, collaborative, customizable, and cross-platform applications. It enables the creation of compound documents, which are created and edited by several cooperating applications working within a single document. OpenDoc also supports scripting and extension mechanisms that allow for communication among parts of a document. OpenDoc allows for multiple parts in a single document. These different parts can range from a spread sheet to voice inside of the document.

OpenDoc consists of a set of shared libraries that can be used to build editors and viewers for compound documents, as well as other compound software to provide services to documents. This book provides the necessary public methods for programmers of document software to support voice, multimedia, spread sheets, charts, and graphics in their documents.

---

## Class Descriptions

Class descriptions in this book are in alphabetical order. Each class description states how the class is to be used-whether the class can be subclassed, how objects of the class are created, and how a part can use an object of the class. If a class is not used directly by parts but only by the OpenDoc document shell or a container application, the class description makes this restriction clear.

---

## Abstract Superclasses

This book and other OpenDoc manuals use the term *abstract superclass* to describe a class, such as [ODPersistentObject](#), that must be subclassed rather than used directly. In some cases, a superclass is not abstract, but implements basic functionality that can be further



extended through subclassing. In the case of [ODTransform](#), for example, parts can create and use objects of the superclass and need not implement subclasses.

---

## Inheritance Relationships

The SOMObject class is the root of the OpenDoc class hierarchy. [OpenDoc Class Hierarchy](#) contains an illustration showing the entire class hierarchy for OpenDoc. In this book, each class description shows the location of the class in the OpenDoc class hierarchy by listing its ancestors and its subclasses.

The ancestors of a given class are shown as a path down the class hierarchy, beginning with the SOMObject class and ending with the class being described. For example, the following branch appears in the class description of the [ODDocument](#) class. The branch indicates that [ODRefCntObject](#) is a superclass of [ODDocument](#), [ODObject](#) is the superclass of [ODRefCntObject](#), and SOMObject is the superclass of [ODObject](#).

```
SOMObject
  ODObject
    ODRefCntObject
      ODDocument
```

The subclasses of a given class are not shown; if you are interested in the descendant classes in the hierarchy, you can refer to the descriptions of the illustration in the section [OpenDoc Class Hierarchy](#).

---

## Method Descriptions

For each OpenDoc class, descriptions of its public methods follow the general class description. Methods are described in alphabetic order. If a method is not used directly by parts but only by the OpenDoc document shell or a container application, the method description makes this restriction clear.

Because method names are not necessarily unique, cross-references to method descriptions in this book use a fully qualified method name with the notation *ClassName::MethodName*. For example, [ODDocument::CreateDraft](#) means the [CreateDraft](#) method of the [ODDocument](#) class.

---

## "Does" Versus "Should"

Most method descriptions in this book state what the method actually does. If a method can be overridden, however, its description states what the override method *should* do. If you override the method, you should implement your override method to behave as the description says it should. If you call such a method, you can assume that the method behaves as it should, but you should be aware that the actual behavior depends on how each override method is implemented.

---

## Object References

OpenDoc objects are always passed by reference when they are used as parameters to methods or as values returned from methods. To emphasize this fact, this book uses the term *reference* exclusively to mean *reference to an object*. The way in which objects are referenced may vary among programming languages. If you use the C++ language, note that the use of the term *reference* in this book does not necessarily imply a C++ reference.

---

## "This" Object

Because OpenDoc is an object-oriented class library, multiple objects of a given OpenDoc class can be created at run time. For this reason, method descriptions use the term *this object* (for instance, "this part" or "this frame") to refer to the specific object whose method is being called.

---

## Override Information

If a method must be or cannot be overridden, the method description makes this restriction clear. If an override method must or cannot call its inherited method, the method description makes this restriction clear. Otherwise, the following default information can be assumed for each method where overriding information is not explicitly stated.

If you subclass *ClassName*, you can override this method. Your override method can call its inherited method at any point in your implementation (it does not matter where).

---

## OpenDoc and SOMObjects

OpenDoc objects follow the System Object Model (SOM), an object-oriented programming technology for building class libraries that support object binding at run time.

The interface to SOM classes must be written in the CORBA Interface Definition Language (IDL), a programming-language-neutral syntax for creating interfaces. The interfaces are compiled separately from the implementations of the classes by the SOMObjects IDL compiler, which supports object-oriented programming languages, such as C++, and procedural programming languages, such as C.

Because OpenDoc uses SOMObjects and IDL, part editors and other OpenDoc classes that have been created with different compilers or in different programming languages can nevertheless communicate properly with one another. Furthermore, they can be independently revised and extended and still work together. Method prototypes in the IDL syntax is similar to that of C and C++.

---

## IDL Prototypes

Method prototypes in the IDL syntax is similar to that of C and C++. IDL includes essentially the same character set, whitespace rules, comment styles, preprocessing capabilities, identifier-naming rules, and rules for literals. But there are a few notable differences in source-code appearance when declaring or calling methods of SOM-based objects:

- In IDL method declarations, each parameter declaration is preceded by a *directional attribute* (in, out, or inout) that notes whether the parameter is used as an input, a result, or both.
- The C++ interface to any method of a SOM object includes an extra initial parameter, the environment parameter (ev), used by all SOM methods to pass exceptions. See the chapter on OpenDoc run-time features in the *OpenDoc Programming Guide* for information on SOM exception handling.
- The C interface to any SOM method includes another extra parameter (somSelf) before the environment parameter, specifying the object to which the method call is directed.

As an example of IDL syntax, here is the prototype for the [AcquireFrame](#) method of the [ODDraft](#) class:

```
ODFrame AcquireFrame (in ODStorageUnitID id);
```

The directional attribute in indicates that the *id* parameter is only passed into the method.

The SOMObjects IDL compiler converts IDL declarations into declarations and stub definitions in the selected implementation language. It adds any necessary parameters and converts out and inout parameters appropriately for the selected language. For example, out parameters may be implemented as pointers.

---

## SOM Development

All OpenDoc objects are SOM objects, descended from the class SOMObject. Your subclass of [ODPart](#) must likewise be a SOM class. If you want other classes you define to be SOM classes, then you must write your interfaces in IDL, separate from your implementations. You must compile your interfaces with the SOMObjects IDL compiler, which can produce header files and stub implementation source files in the various programming languages supported by the SOMObjects IDL compiler. Options to the compiler specify which files to produce. You

complete your development by writing your implementations into the stub implementation files and compiling them, along with the header files, using a standard compiler for your programming language.

For a more detailed description of the Interface Definition Language and instructions on programming with SOM, see the *System Object Model Guide and Reference*.

---

## Conventions Used in This Book

This book uses various conventions to present certain types of information.

---

## Documentation Conventions

Throughout this library of documents, the following conventions distinguish the different elements of text:

plain text	Function names, structure names, data types names, message names, enumerated types, and constant names.
Initial capitalization	Key names, push buttons, checkboxes, radio buttons, group-box controls, drop-down list box, dialog windows, spin buttons, combo-boxes, SLE and MLE fields.
CAPITALS	File names and error codes.
monospace	Programming examples and user input at the command line prompt or into an entry field.
<b>bold</b>	Action bar choices and menu items.
<i>italics</i>	Parameters, structure fields, titles of documents, and first occurrences of words with special meaning.

---

## OS/2 Information

OpenDoc is a cross-platform technology, and most of its concepts and features are platform-independent. Thus, even though this book is specifically designed for OS/2 developers, the information is organized and presented in as platform-neutral as possible.

Throughout this book, any methods, function, macros, types and constants that are specific to OS/2 are called out as such. That way, you can get a general idea of how platform-specific your code must be, and therefore, how simple or complex it may be to convert it to another platform.

---

## Conventions Used in Function Descriptions

The following figure illustrates a sample function:

```
SampleFunctionName
    A description of the function.

#define INCL_OSA
#include <os2.h>
```

```
dat_return SampleFunctionName (dat_1 param_1, ..., dat_n param_n)
```

**Parameter** **param\_1** (dat\_1) - parameter type      A description of param\_1.      .      .      .      **param\_n** (dat\_n) - parameter type  
A description of param\_n.      **Returns** **param\_return** (dat\_return) - returns      A description of param\_return.      **Remarks**  
Information about using the function.      **Related Functions**      A list of related function described in this book.      **Example Code**      An  
example of a code segment that uses the function.

The following list explains the purpose of each item on a reference page.

#### Heading

The heading shows the function name. Functions are separated by a rule above each name.

#### Description

The description follows the heading and explains what the function does. It also identifies options and requirements for calling the function.

#### Syntax

The syntax describes the C-language calling syntax of the function. This includes any #define and #include directives that must be used when calling this function.

#### Parameters

This section lists each parameter passed by the function and provides its data type, parameter type, and a brief description. All parameters must be specified when calling the function.

All data types are written in uppercase and lowercase to match the header files. Generally, a data type of "Pxxxxxx" implicitly defines a pointer to the data type "xxxxxx".

The terms KODNULL and KODNULLID applied to a parameter indicates the presence of a parameter that has no value.

There are three parameter types:

Input	A value that is specified by the programmer, but is not modified by the function. If the parameter is a pointer to a value, neither the pointer nor the information it points to changes.
Output	A parameter that receives the function's output. If the value represents a pointer, the pointer remains the same, but the information it points to changes.
Input/Output	A value that is provided as input to the function and whose content is changed upon return. If the parameter is a pointer to a value, the pointer remains the same, but the information it points to changes.

#### Returns

This section includes a list of possible return codes or values.

#### Remarks

This section contains additional information about the function, such as:

- guidelines for specifying parameters
- instructions about the order in which to use the function with related functions
- when you might want to use one function in place of another.

#### Related Functions

This list shows other functions (if any) defined in this document that are related to the function being described.

#### Example Code

This section provides a programming example of how to use the function in an OpenDoc part or application program. The example uses valid and realistic values for parameters.

In the online version of this book, you can copy the example to your application source file.

---

## Conventions Used in Method Descriptions

The following figure illustrates a sample method:

## SampleMethodName

A description of the method.

```
#define INCL_OSA
#include <os2.h>
```

```
dat_return SampleMethodName (dat_1 param_1, ..., dat_n param_n)
```

**Parameter** **param\_1** (dat\_1) - parameter type A description of param\_1. . . . **param\_n** (dat\_n) - parameter type A description of param\_n. **Returns** **param\_return** (dat\_return) - returns A description of param\_return. **Remarks** Information about using the method. **Exception Handling** Information about exceptions (error codes). **Override Policy** Information about whether this method can be overridden. **Related Methods** A list of related method described in this book. **Example Code** An example of a code segment that uses the method.

The following list explains the purpose of each item on a reference page.

### Heading

The heading shows the method name. Methods are separated by a rule above each name.

### Description

The description follows the heading and explains what the method does. It also identifies options and requirements for calling the method.

### Syntax

The method syntax describes the C++-language calling syntax of the method. This includes any #define and #include directives that must be used when calling the methods.

### Parameters

This section lists each parameter passed by the method and provides its data type, parameter type, and a brief description. All parameters must be specified when calling the method.

All data types are written in uppercase and lowercase to match the header files. Generally, a data type of "Pxxxxxx" implicitly defines a pointer to the data type "xxxxxx".

The terms KODNULL and KODNULLID applied to a parameter indicates the presence of a parameter that has no value.

There are three parameter types:

Input	A value that is specified by the programmer, but is not modified by the method. If the parameter is a pointer to a value, neither the pointer nor the information it points to changes.
Output	A parameter that receives the method's output. If the value represents a pointer, the pointer remains the same, but the information it points to changes.
Input/Output	A value that is provided as input to the method and whose content is changed upon return. If the parameter is a pointer to a value, the pointer remains the same, but the information it points to changes.

The C++ interface to any method of a SOM object includes an extra initial parameter, the environment parameter (ev), used by all SOM methods to pass exceptions. See the chapter on OpenDoc run-time features in the *OpenDoc Programming Guide* for information on SOM exception handling. The C interface to any SOM method includes another extra parameter (somSelf) before the environment parameter, specifying the object to which the method call is directed. For example, the syntax of the [ODPart::CanvasUpdated](#) method in the SOM language is as follows:

```
void CanvasUpdated (ODPart *somSelf, Environment *ev, ODCanvas *canvas);
```

If you are using the C++ language, the syntax is as follows:

```
void CanvasUpdated (Environment *ev, ODCanvas *canvas);
```

### Returns

This section includes a list of possible return codes or values.

### Remarks

This section contains additional information about the method such as:

- guidelines for specifying parameters
- instructions about the order in which to use the method, with related methods
- when you might want to use one method in place of another.

#### Exception Handling

This section describes the exceptions (or error codes) for each method. The exception handling section is not shown if there are no exceptions for the method.

#### Override Policy

This section describes whether the method can be overridden. If a method must or cannot be overridden, this method description makes this restriction clear. If an override method must or cannot call its inherited method, the method description makes this restriction clear. Otherwise, the following default information can be assumed for each method where overriding information is not explicitly stated.

If you subclass *ClassName*, you can override this method. Your override method can call its inherited method at any point in your implementation (it does not matter where).

#### Related Methods

This list shows other methods (if any) defined in this document that are related to the method being described.

#### Example Code

This section provides a programming example of how to use the method in an application program. The example uses valid and realistic values for parameters.

In the online version of this book, you can copy the example to your application source file.

-----

## Programming Considerations

This section provides information you need to consider before you begin calling OpenDoc functions and methods.

-----

## Header Files

All functions and methods require a "#include" statement for the system header file OS2.H:

```
#include <os2.h>
```

Most functions and methods also require a "#define" statement to select an appropriate (conditional) section of the header file and, hence, the required prototype. Where this is necessary, it is shown at the head of the function definition in the form:

```
#define INCL_name
```

**Note:** These "#define" statements must precede the "#include <os2.h>" statement.

-----

## Implicit Pointer Data Types

A data-type name beginning with "P" is a pointer to data of another type, unless defined otherwise in the header files. For example, PPartKindInfo is a pointer to data of type [PartKindInfo](#).

In the data-type summary, [Data Types](#), no explicit "typedefs" are shown for pointers; therefore, if no data-type definition can be found in the summary for a data-type name "Pxxxxxx", it represents a pointer to the data type "xxxxxx", for which a definition should be found in the

reference.

The implicit type definition needed for such a pointer "Pxxxxx" is:

```
typedef xxxxxx *Pxxxxxx;
```

Such definitions are provided in the header files.

---

## Storage Mapping of Data Types

The storage mapping of data types is dependent on the machine architecture. To be portable, applications must access data using the type definitions supplied for the environment in which they execute.

---

## Message Queues

Usually, when an application thread calls a Presentation Manager (PM) function, a message queue must be available for that thread. This means that, before calling the function, WinCreateMsgQueue must be called by the same thread.

It is recommended that you create a message queue for every thread that calls an OpenDoc method, because a PM function might be used by the methods you are calling.

---

## Stack Size

Existing 16-bit applications (small and tiny models) must have a 4KB stack available when they enter system calls; otherwise, the stack can overflow into the data area.

---

## C++ Considerations

This section contains several topics you should take into consideration if you are using C++.

---

## C++ Header Files

Several of the typedefs have been changed in the C++ header files. For example, `unsigned char` in the C header files are `char` in the C++ header files:

```
typedef unsigned char BYTE;
```

has changed to

```
typedef char BYTE;
```

The existing sample programs that are included in the IBM Developer's Toolkit for OS/2 Warp (Warp Toolkit) can be used with either set of the

header files.

---

## PCSZ Data Type

OS/2 functions with parameters for type [PSZ](#) that do not modify the contents of the passed string, have been updated in the C++ header files as parameters of type [PCSZ](#). The use of [PCSZ](#) data types allows for better optimization by the compiler and is more semantically compatible with C++. Existing code that calls functions with parameters of type [PSZ](#) continues to work correctly.

**Note:** The [PCSZ](#) data type is defined in the C++ header files included with this product. The use of the "const" keyword is not necessarily specific to C++. Certain C compilers support it as well.

A smaller, faster executable is often produced if the data item passed in a parameter list is declared as "const".

If the data item is declared as "const", then it must not be changed by the function.

If a function has a string as a parameter that is not changed by the function, the string parameter can be declared as a "const" string, or a [PCSZ](#). [PCSZ](#) is defined in the C++ header files as a "const" pointer to a nullL-delimited string. The "const" means that the function does not change the contents of the string. The compiler simply passes a pointer to the string in the function parameter list. If the parameter is declared as a normal [PSZ](#) (not "const"), the compiler assumes that the function might change the string. Under these circumstances, the compiler adds code to make a copy of the string and then passes a pointer to the copy, rather than passing a pointer to the original string.

---

## LINK386

The C++ compiler provides a dynamic link library to be used by LINK386 when generating error messages. This DLL converts a compiler-generated mangled name into the function prototype. If the DLL is not present, an error message is displayed, and LINK386 displays the compiler-generated mangled name in error messages.

---

## Double-Byte Character Set (DBCS)

Throughout this publication, you will see references to specific values for character strings. The values are for single-byte character set (SBCS). If you use the double-byte character set (DBCS), note that one DBCS character equals two SBCS characters.

---

## Related Publications

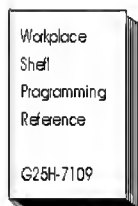
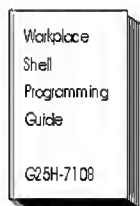
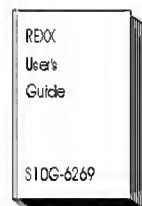
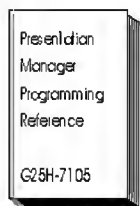
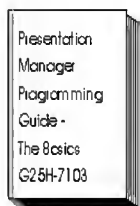
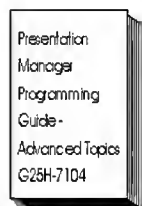
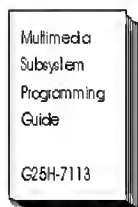
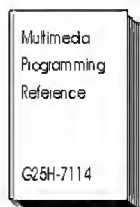
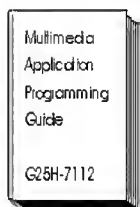
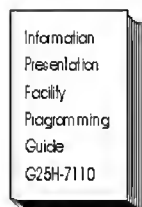
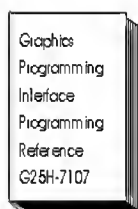
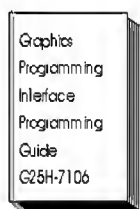
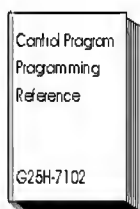
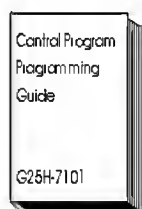
The following diagram provides an overview of the OS/2 Warp Version 3, Technical Library.

Books can be ordered by calling toll free 1-800-879-2755 weekdays between 8:00 a.m. and 8:00 p.m. (EST). In Canada, call 1-800-465-1234.

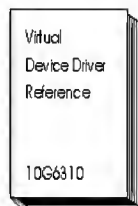
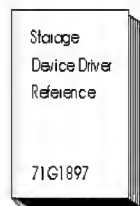
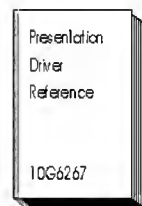
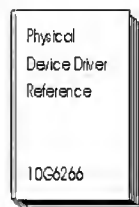
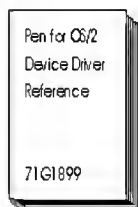
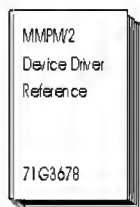


## OS/2 Version 3 Technical Library

### G25H - 7116



### IBM Device Driver Publications for OS/2



---

## OpenDoc Classes and Methods

These chapters contain an alphabetic listing of the OpenDoc object classes and their methods. These sections contain technical reference information. See the *OpenDoc Programming Guide* for OpenDoc guide information. For information on the System Object Model (SOM), see the *System Object Model Guide and Reference* .

The following list describes some terminology used in the following sections:

class	A way of categorizing objects based on their behavior and shape. A class is, in effect, a definition of a generic object. In SOM, a class is a special kind of object that can manufacture other objects that all have a common shape and exhibit similar behavior (more precisely, all of the objects manufactured by a class have the same memory layout and share a common set of methods). New classes can be defined in terms of existing classes through a technique known as <i>inheritance</i> .
class method	A class method of class <X> is a method provided by the metaclass of class <X>. Class methods are executed without requiring any instances of class <X> to exist and are frequently used to create instances.
inheritance	The technique of specifying the shape and behavior of one class (called a <i>subclass</i> ) as incremental differences from another class (called the <i>parent class</i> or <i>superclass</i> ). The subclass inherits the superclass' state representation and methods and can provide additional data elements and methods. The subclass also can provide new functions with the same method names used by the superclass. Such a subclass method is said to override the superclass method and will be selected automatically by method resolution on subclass instances. An overriding method can elect to call upon the superclass' method as part of its own implementation.
instance	(Or object instance). A specific object, as distinguished from the abstract definition of an object referred to as its class.
instance method	A method that is valid for a particular object.
metaclass	A class whose instances are all classes. In SOM, any class descended from SOMClass is a metaclass. The methods of a metaclass are sometimes called "class" methods.
method	One of the units that make up the behavior of an object. A method is a combination of a function and a name, such that many different functions can have the same name. Which function the name refers to, at any point, depends on the object that is to execute the method and is the subject of method resolution.
object	<p>The elements of data and function that programs create, manipulate, pass as arguments, and so forth. An object is a way of associating specific data values with a specific set of named functions (called <i>methods</i> ) for a period of time (referred to as the <i>lifetime</i> of the object). The data values of an object are referred to as its <i>state</i> . In SOM, objects are created by other objects called <i>classes</i> . The specification of what comprises the set of functions and data elements that make up an object is referred to as the <i>definition</i> of a class.</p> <p>SOM objects offer a high degree of <i>encapsulation</i> . This property permits many aspects of the implementation of an object to change without affecting client programs that depend on the object's behavior.</p>
object class	See <i>class</i> .
object instance	See <i>instance</i> .
subclass	A class that inherits from another class. See <i>inheritance</i> .
superclass	A class from which another class inherits. See <i>inheritance</i> .

---

## OpenDoc Class Hierarchy

The following figure lists the predefined OpenDoc object classes in a hierarchical order. SOMObject is the root class for all OpenDoc object classes. Each branch in the tree represents an immediate descendant (subclass) of an OpenDoc object class. The figure also shows the name of the class definition files (IDL) for each OpenDoc class.

**CLASS NAME**

**CLASS DEFINITION FILE**

SOMObject	somobj.idl
SOMClassMgr	somcm.idl
SOMClass	somcls.idl
ODObject	odobject.idl
ODArbitrator	arbitrat.idl
ODBaseCanvas	canvasb.idl
ODCanvas	canvas.idl
ODBaseClipboard	clipbdb.idl
ODClipboard	clipbd.idl
ODBaseDispatcher	disptchb.idl
ODDispatcher	disptch.idl
ODBaseDragAndDrop	dragdrpb.idl
ODDragAndDrop	dragdrp.idl
ODBaseDragItemIterator	dgitmitb.idl
ODDragItemIterator	dgitmit.idl
ODBaseFacet	facetb.idl
ODFacet	facet.idl
ODBaseLinkSpec	linkspcb.idl
ODLink	linkspec.idl
ODBaseMessageInterface	mssgintb.idl
ODMessageInterface	mssgintf.idl
ODBaseNameResolver	namrslvb.idl
ODNameResolver	namrslvr.idl
ODBaseNameSpaceManager	nmspcmgb.idl
ODNameSpaceManager	nmspcmg.idl
ODBaseSession	odsessnb.idl
ODSession	odsessn.idl
ODBaseStorageSystem	odstorb.idl
ODStorageSystem	odstor.idl
ODBaseTranslation	transltb.idl
ODTranslation	translt.idl
ODBaseWindowState	winstatb.idl
ODWindowState	winstat.idl
ODBinding	odbindng.idl
ODDispatchMode	dispmod.idl
ODDispatcher	disptch.idl
ODStandardDispatchModule	stddisp.idl
ODEmbeddedFramesIterator	embfritr.idl
ODFacetIterator	facetitr.idl
ODFocusModule	focusmod.idl
ODFocusOwnerIterator	focusown.idl
ODFocusSet	focusset.idl
ODFocusSetIterator	focusitr.idl
ODFrameFacetIterator	frfaiitr.idl
ODHelp	odhelp.idl
ODInfo	info.idl
ODNameSpace	namspac.idl
ODObjectNameSpace	valuens.idl
ODValueNameSpace	objectns.idl
ODObjectIterator	objctitr.idl
ODPlatformTypeList	pftypls.idl
ODPlatformTypeListIterator	pftlitr.idl
ODRefCntObject	refctobj.idl
ODBaseMenuBar	menubarb.idl
ODMenuBar	menubar.idl
ODPopup	popup.idl
ODBaseShape	shapeb.idl
ODShape	shape.idl
ODBaseTransform	trnsfrmb.idl
ODTransform	trnsform.idl
ODBaseWindow	windowb.idl
ODWindow	window.idl
ODContainer	odctr.idl
ODDocument	document.idl
ODDraft	draft.idl
ODExtension	extensn.idl
ODBaseSemanticInterface	semtintb.idl
ODSemanticInterface	semtintf.idl
ODSettingsExtension	settings.idl
ODStatusLineExtension	status.idl

ODViewExtension	odviewex.idl
ODPersistentObject	pstobj.idl
ODBaseLink	linkb.idl
ODLink	link.idl
ODBaseLinkSource	linksrcb.idl
ODLinkSource	linksrc.idl
ODFrame	frame.idl
ODPart	part.idl
ODStorageUnit	storageu.idl
ODStorageUnitCursor	sucursor.idl
ODStorageUnitRefIterator	surefitr.idl
ODStorageUnitView	suview.idl
ODTemplates	odtemps.idl
ODTypeList	typelist.idl
ODTypeListIterator	typlsitr.idl
ODUndo	undo.idl
ODValueIterator	valueitr.idl
ODWindowIterator	winiter.idl
ODPartHandlerInfo	partinfo.idl
ODPartHandlerRegistry	odprtreg.idl
RegistryManager	rmanager.idl

Some OpenDoc object classes cannot be instantiated—that is, you cannot create an instance for that object class. These classes are base classes that provide support for descendant classes that can be instantiated.

## ODAddressDesc

**Class Definition File:** ODADDRDES.IDL

### Class Hierarchy

```

SOMObject
  ODObject
    ODDesc
      ODAddressDesc

```

### Description

An object of the ODAddressDesc class is a wrapper for an *address descriptor structure*, a descriptor structure of type AAddressDesc that contains a target destination address. Every OSA event includes an attribute specifying the destination address of the target application.

For more information on OSA events and the AAddressDesc type, see the chapter introducing OSA events in the *Open Scripting Architecture Guide and Reference for OS/2*. For general information on scripting support in OpenDoc, see the chapter on semantic events and scripting in the *OpenDoc Programming Guide*.

### Methods

The methods defined by the ODAddressDesc class include:

- [InitODAddressDesc](#)

### Overridden Methods

There are currently no methods overridden by the ODAddressDesc class.

## InitODAddressDesc

## InitODAddressDesc - Syntax

This method creates and initializes an address descriptor.

```
#define INCL_ODADDRESSDESC
#define INCL_ODAPI
#include <os2.h>
```

```
InitODAddressDesc();
```

---

## InitODAddressDesc - Return Value

None.

---

## InitODAddressDesc - Parameters

None.

---

## InitODAddressDesc - Remarks

There is no factory method for the [ODAddressDesc](#) class; after creating a new address descriptor object, OpenDoc or your part must call this method to initialize the new address descriptor object.

---

## InitODAddressDesc - Topics

### Class:

[ODAddressDesc](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

---

## ODArbitrator

**Class Definition File:** ARBITRAT.IDL

### Class Hierarchy

SOMObject

## Description

An object of the ODArbitrator class manages temporary ownership of shared resources or features among parts.

A *focus* is a designation of ownership of a given shared resource. Foci are defined according to the type of resource that each represents. For example, focus types include the keystroke focus, menu bar focus, and selection focus. Other foci can be associated with system-wide resources, such as serial ports. Foci are owned by frames. A frame's part relinquishes ownership of foci when asked to, such as when another part is requesting ownership or when a frame is deleted.

When a document is opened, the session object creates a single arbitrator object. All parts of that document share the arbitrator object; you can obtain a reference to it by calling the session object's [GetArbitrator](#) method.

The OpenDoc arbitrator keeps track of which part owns a focus by consulting the focus module for that focus. A focus is registered if it has a focus module. The arbitrator is used by the OpenDoc dispatcher to determine where to send events. For example, the dispatcher, which is responsible for distributing events to part editors, directs keyboard events to the part that currently owns the keystroke focus and menu events to the part that currently owns the menu bar focus.

Foci may be exclusive or nonexclusive. All of the standard foci defined by OpenDoc are *exclusive*, meaning that only one frame at a time can own a focus. The ODArbitrator class also supports *nonexclusive* foci, which can be owned by several frames. In such cases, the arbitrator provides a central location in which you can record and later obtain a list of the owners for that focus.

For more information related to foci, focus sets, focus types, and focus modules, see the class descriptions for [ODFocusModule](#), [ODFocusOwnerIterator](#), [ODFocusSet](#), and [ODFocusSetIterator](#). For more information related to the dispatcher, see the class description for [ODDispatcher](#).

## Methods

The methods defined by the ODArbitrator class include:

- [AcquireFocusOwner](#)
- [CreateFocusSet](#)
- [CreateOwnerIterator](#)
- [GetFocusModule](#)
- [IsFocusExclusive](#)
- [IsFocusRegistered](#)
- [RegisterFocus](#)
- [RelinquishFocus](#)
- [RelinquishFocusSet](#)
- [RequestFocus](#)
- [RequestFocusSet](#)
- [TransferFocus](#)
- [TransferFocusSet](#)
- [UnregisterFocus](#)

## Overridden Methods

There are currently no methods overridden by the ODArbitrator class.

---

# AcquireFocusOwner

---

## AcquireFocusOwner - Syntax

This method returns a reference to the frame that owns the specified exclusive focus.

```
#define INCL_ODARBITRATOR
#define INCL_ODAPI
#include <os2.h>

ODTypeToken    focus;
ODFrame        *rc;
```

```
rc = AcquireFocusOwner(focus);
```

---

## AcquireFocusOwner Parameter - focus

**focus** (ODTypeToken) - input

A tokenized string representing the focus type to be acquired.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the clipboard focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus

The frame with menu focus receives menu events.

kODModalFocus

A frame with modal focus is notifying other frames that it is the only currently modal frame.

kODMouseFocus

The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

kODScrollingFocus

The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

kODSelectionFocus

The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

---

## AcquireFocusOwner Return Value - rc

**rc** (ODFrame \*) - returns

A reference to the frame that owns the specified exclusive focus or kODNULL if the focus is not owned by any frame.

---

## AcquireFocusOwner - Parameters

**focus** (ODTypeToken) - input

A tokenized string representing the focus type to be acquired.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the clipboard focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus

The frame with menu focus receives menu events.

kODModalFocus

A frame with modal focus is notifying other frames that it is the only currently modal frame.

kODMouseFocus

The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

kODScrollingFocus

The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

kODSelectionFocus

The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

**rc** (ODFrame \*) - returns

A reference to the frame that owns the specified exclusive focus or kODNULL if the focus is not owned by any frame.

-----

## AcquireFocusOwner - Remarks

A part can obtain a reference to the owner of a specified exclusive focus by calling this method. This method looks up the focus module for the given focus for the specified focus and calls the focus module's [AcquireFocusOwner](#) method. If the focus is not registered, the focus has no focus module and this method is never called.

This method increments the reference count of the returned frame object. When you have finished using that frame object, you should call its [Release](#) method.

-----

## AcquireFocusOwner - Related Methods

### Related Methods

- [ODFocusModule::AcquireFocusOwner](#)
- [ODSession::Tokenize](#)

-----

## AcquireFocusOwner - Topics

### Class:

ODArbitrator

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

-----

## CreateFocusSet

-----

## CreateFocusSet - Syntax

This method creates and initializes a focus set.



```
#define INCL_ODARBITRATOR
#define INCL_ODAPI
#include <os2.h>
```

```
ODFocusSet      *rc;

rc = CreateFocusSet();
```

-----

## CreateFocusSet Return Value - rc

**rc** (ODFocusSet \*) - returns  
A reference to a newly created focus set.

-----

## CreateFocusSet - Parameters

**rc** (ODFocusSet \*) - returns  
A reference to a newly created focus set.

-----

## CreateFocusSet - Remarks

Your part calls this method to create a focus set to pass to the arbitrator's [RequestFocusSet](#) method.

-----

## CreateFocusSet - Exception Handling

kODErrOutOfMemory

There is not enough memory to allocate the focus set object.

-----

## CreateFocusSet - Related Methods

### Related Methods

- [ODArbitrator::RequestFocusSet](#)

-----

## CreateFocusSet - Topics

**Class:**

Select an item:

[Syntax](#)[Parameters](#)[Returns](#)[Remarks](#)[Exception Handling](#)[Related Methods](#)

---

# CreateOwnerIterator

---

## CreateOwnerIterator - Syntax

This method creates a focus-owner iterator to give callers accesses to the frames that own a nonexclusive focus.

```
#define INCL_ODARBITRATOR
#define INCL_ODAPI
#include <os2.h>

ODTypeToken          focus;
ODFocusOwnerIterator *rc;

rc = CreateOwnerIterator(focus);
```

---

## CreateOwnerIterator Parameter - focus

**focus** ([ODTypeToken](#)) - input

A tokenized string representing the focus type whose owners are to be accessed.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the clipboard focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus

The frame with menu focus receives menu events.

kODModalFocus

A frame with modal focus is notifying other frames that it is the only currently modal frame.

kODMouseFocus

The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

kODScrollingFocus

The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

kODSelectionFocus

The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

---

# CreateOwnerIterator Return Value - rc

**rc** (ODFocusOwnerIterator \*) - returns

A reference to a newly created focus-owner iterator or kODNULL if the specified focus is exclusive.

---

## CreateOwnerIterator - Parameters

**focus** (ODTypeToken) - input

A tokenized string representing the focus type whose owners are to be accessed.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the clipboard focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus

The frame with menu focus receives menu events.

kODModalFocus

A frame with modal focus is notifying other frames that it is the only currently modal frame.

kODMouseFocus

The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

kODScrollingFocus

The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

kODSelectionFocus

The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

**rc** (ODFocusOwnerIterator \*) - returns

A reference to a newly created focus-owner iterator or kODNULL if the specified focus is exclusive.

---

## CreateOwnerIterator - Remarks

To get access to the owner of an exclusive focus, use the arbitrator's [AcquireFocusOwner](#) method.

While you are using a focus-owner iterator, you should not modify the list of focus owners. You must postpone adding items to or removing items from the list of focus owner until after you have deleted the iterator.

---

## CreateOwnerIterator - Exception Handling

kODErrFocusNotRegistered

The requested focus is not registered.

kODErrOutOfMemory

There is not enough memory to allocate the focus-owner iterator object.

---

## CreateOwnerIterator - Related Methods

## Related Methods

- [ODArbitrator::AcquireFocusOwner](#)
- [ODSession::Tokenize](#)

---

# CreateOwnerIterator - Topics

## Class:

ODArbitrator

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

# GetFocusModule

---

## GetFocusModule - Syntax

This method returns a reference to the focus module for the specified focus.

```
#define INCL_ODARBITRATOR
#define INCL_ODAPI
#include <os2.h>

ODTypeToken      focus;
ODFocusModule    *rc;

rc = GetFocusModule(focus);
```

---

## GetFocusModule Parameter - focus

**focus** ([ODTypeToken](#)) - input

A tokenized string representing the focus type to be acquired.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the clipboard focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus	The frame with menu focus receives menu events.
kODModalFocus	A frame with modal focus is notifying other frames that it is the only currently modal frame.
kODMouseFocus	The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.
kODScrollingFocus	The frame with the scrolling focus receives page-movement key events (such as <b>PageUp</b> ). This frame does not need to own keyboard focus.
kODSelectionFocus	The frame with the selection focus receives modified mouse-click events ( <b>Shift</b> +click and <b>Ctrl</b> +click). OpenDoc draws the active frame border around all facets of this frame.

-----

## GetFocusModule Return Value - rc

**rc** (ODFocusModule \*) - returns  
A reference to a focus module for the specified focus.

-----

## GetFocusModule - Parameters

**focus** (ODTypeToken) - input  
A tokenized string representing the focus type to be acquired.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus	The frame with the <a href="#">clipboard</a> focus has access to the clipboard.
kODKeyFocus	The frame with keyboard focus receives keyboard events (excluding page-movement key events).
kODMenuFocus	The frame with menu focus receives menu events.
kODModalFocus	A frame with modal focus is notifying other frames that it is the only currently modal frame.
kODMouseFocus	The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.
kODScrollingFocus	The frame with the scrolling focus receives page-movement key events (such as <b>PageUp</b> ). This frame does not need to own keyboard focus.
kODSelectionFocus	The frame with the selection focus receives modified mouse-click events ( <b>Shift</b> +click and <b>Ctrl</b> +click). OpenDoc draws the active frame border around all facets of this frame.

**rc** (ODFocusModule \*) - returns  
A reference to a focus module for the specified focus.

-----

## GetFocusModule - Related Methods

### Related Methods

- [ODSession::Tokenize](#)

---

# GetFocusModule - Topics

## Class:

ODArbitrator

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Related Methods](#)

---

## IsFocusExclusive

---

### IsFocusExclusive - Syntax

This method indicates whether the specified focus is exclusive.

```
#define INCL_ODARBITRATOR
#define INCL_ODAPI
#include <os2.h>

ODTypeToken    focus;
ODBoolean      rc;

rc = IsFocusExclusive(focus);
```

---

### IsFocusExclusive Parameter - focus

**focus** ([ODTypeToken](#)) - input

A tokenized string representing the focus type to be tested.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the clipboard focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus

The frame with menu focus receives menu events.

kODModalFocus

A frame with modal focus is notifying other frames that it is the only currently modal frame.

kODMouseFocus

The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

kODScrollingFocus

The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

kODSelectionFocus

The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

---

## IsFocusExclusive Return Value - rc

**rc** (ODBoolean) - returns

A flag indicating whether the specified focus is exclusive.

kODTrue

The specified focus is exclusive.

kODFalse

The specified focus is nonexclusive.

---

## IsFocusExclusive - Parameters

**focus** (ODTypeToken) - input

A tokenized string representing the focus type to be tested.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the clipboard focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus

The frame with menu focus receives menu events.

kODModalFocus

A frame with modal focus is notifying other frames that it is the only currently modal frame.

kODMouseFocus

The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

kODScrollingFocus

The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

kODSelectionFocus

The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

**rc** (ODBoolean) - returns

A flag indicating whether the specified focus is exclusive.

kODTrue

The specified focus is exclusive.

kODFalse

The specified focus is nonexclusive.

---

## IsFocusExclusive - Exception Handling

KODErrFocusNotRegistered

The requested focus is not registered.

---

## IsFocusExclusive - Related Methods

## Related Methods

- [ODSession::Tokenize](#)

---

# IsFocusExclusive - Topics

## Class:

ODArbitrator

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Exception Handling](#)

[Related Methods](#)

---

# IsFocusRegistered

---

## IsFocusRegistered - Syntax

This method indicates whether the specified focus is registered.

```
#define INCL_ODARBITRATOR
#define INCL_ODAPI
#include <os2.h>

ODTypeToken    focus;
ODBoolean      rc;

rc = IsFocusRegistered(focus);
```

---

## IsFocusRegistered Parameter - focus

**focus** ([ODTypeToken](#)) - input

A tokenized string representing the focus type to be tested.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the clipboard focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus

The frame with menu focus receives menu events.



kODModalFocus	A frame with modal focus is notifying other frames that it is the only currently modal frame.
kODMouseFocus	The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.
kODScrollingFocus	The frame with the scrolling focus receives page-movement key events (such as <b>PageUp</b> ). This frame does not need to own keyboard focus.
kODSelectionFocus	The frame with the selection focus receives modified mouse-click events ( <b>Shift</b> +click and <b>Ctrl</b> +click). OpenDoc draws the active frame border around all facets of this frame.

-----

## IsFocusRegistered Return Value - rc

**rc** ([ODBoolean](#)) - returns  
A flag indicating whether the specified focus is registered.

kODTrue	The specified focus is registered.
kODFalse	The specified focus is not registered.

-----

## IsFocusRegistered - Parameters

**focus** ([ODTypeToken](#)) - input  
A tokenized string representing the focus type to be tested.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus	The frame with the clipboard focus has access to the clipboard.
kODKeyFocus	The frame with keyboard focus receives keyboard events (excluding page-movement key events).
kODMenuFocus	The frame with menu focus receives menu events.
kODModalFocus	A frame with modal focus is notifying other frames that it is the only currently modal frame.
kODMouseFocus	The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.
kODScrollingFocus	The frame with the scrolling focus receives page-movement key events (such as <b>PageUp</b> ). This frame does not need to own keyboard focus.
kODSelectionFocus	The frame with the selection focus receives modified mouse-click events ( <b>Shift</b> +click and <b>Ctrl</b> +click). OpenDoc draws the active frame border around all facets of this frame.

**rc** ([ODBoolean](#)) - returns  
A flag indicating whether the specified focus is registered.

kODTrue	The specified focus is registered.
kODFalse	The specified focus is not registered.

-----

## IsFocusRegistered - Remarks

A focus is registered if it has a focus module.

---

## IsFocusRegistered - Related Methods

### Related Methods

- [ODSession::Tokenize](#)
- 

## IsFocusRegistered - Topics

### Class:

ODArbitrator

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## RegisterFocus

---

## RegisterFocus - Syntax

This method adds the specified focus module for the specified focus.

```
#define INCL_ODARBITRATOR
#define INCL_ODAPI
#include <os2.h>

ODTypeToken      focus;
ODFocusModule    *focusModule;

RegisterFocus(focus, focusModule);
```

---

## RegisterFocus Parameter - focus

**focus** ([ODTypeToken](#)) - input

A tokenized string representing the focus type to be assigned to the specified focus module.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus	The frame with the <a href="#">clipboard</a> focus has access to the clipboard.
kODKeyFocus	The frame with keyboard focus receives keyboard events (excluding page-movement key events).
kODMenuFocus	The frame with menu focus receives menu events.
kODModalFocus	A frame with modal focus is notifying other frames that it is the only currently modal frame.
kODMouseFocus	The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.
kODScrollingFocus	The frame with the scrolling focus receives page-movement key events (such as <b>PageUp</b> ). This frame does not need to own keyboard focus.
kODSelectionFocus	The frame with the selection focus receives modified mouse-click events ( <b>Shift</b> +click and <b>Ctrl</b> +click). OpenDoc draws the active frame border around all facets of this frame.

-----

## RegisterFocus Parameter - focusModule

**focusModule** (ODFocusModule \*) - input

A reference to the focus module that is to manage the specified focus or kODNULL to create a standard exclusive focus module.

-----

## RegisterFocus - Return Value

None.

-----

## RegisterFocus - Parameters

**focus** ([ODTypeToken](#)) - input

A tokenized string representing the focus type to be assigned to the specified focus module.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus	The frame with the <a href="#">clipboard</a> focus has access to the clipboard.
kODKeyFocus	The frame with keyboard focus receives keyboard events (excluding page-movement key events).
kODMenuFocus	The frame with menu focus receives menu events.
kODModalFocus	A frame with modal focus is notifying other frames that it is the only currently modal frame.
kODMouseFocus	The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.
kODScrollingFocus	The frame with the scrolling focus receives page-movement key events (such as <b>PageUp</b> ). This frame does not need to own keyboard focus.
kODSelectionFocus	The frame with the selection focus receives modified mouse-click events ( <b>Shift</b> +click and <b>Ctrl</b> +click). OpenDoc

draws the active frame border around all facets of this frame.

**focusModule** (ODFocusModule \*) - input

A reference to the focus module that is to manage the specified focus or kODNULL to create a standard exclusive focus module.

None.

---

## RegisterFocus - Exception Handling

kODErrFocusAlreadyRegistered

The requested focus is already registered.

kODErrOutOfMemory

There is not enough memory to allocate the default focus module.

---

## RegisterFocus - Related Methods

### Related Methods

- [ODSession::Tokenize](#)

---

## RegisterFocus - Topics

### Class:

ODArbitrator

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Exception Handling](#)

[Related Methods](#)

---

## RelinquishFocus

---

## RelinquishFocus - Syntax

This method is called by the current owner of the specified focus to relinquish ownership of it.

```
#define INCL_ODARBITRATOR
#define INCL_ODAPI
#include <os2.h>
```

```
ODTypeToken    focus;
ODFrame        *relinquishingFrame;

RelinquishFocus(focus, relinquishingFrame);
```

## RelinquishFocus Parameter - focus

**focus** (ODTypeToken) - input

A tokenized string representing the focus type to be relinquished.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the clipboard focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus

The frame with menu focus receives menu events.

kODModalFocus

A frame with modal focus is notifying other frames that it is the only currently modal frame.

kODMouseFocus

The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

kODScrollingFocus

The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

kODSelectionFocus

The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

## RelinquishFocus Parameter - relinquishingFrame

**relinquishingFrame** (ODFrame \*) - input

A reference to a frame relinquishing ownership of the focus.

## RelinquishFocus - Return Value

None.

## RelinquishFocus - Parameters

**focus** (ODTypeToken) - input

A tokenized string representing the focus type to be relinquished.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the clipboard focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus

The frame with menu focus receives menu events.

kODModalFocus

A frame with modal focus is notifying other frames that it is the only currently modal frame.

kODMouseFocus

The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

kODScrollingFocus

The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

kODSelectionFocus

The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

**relinquishingFrame** (ODFrame \*) - input

A reference to a frame relinquishing ownership of the focus.

None.

-----

## RelinquishFocus - Remarks

If the focus is exclusive, it has no owner after this method executes successfully.

-----

## RelinquishFocus - Exception Handling

kODErrFocusNotRegistered

The requested focus is not registered.

-----

## RelinquishFocus - Related Methods

### Related Methods

- [ODArbitrator::RelinquishFocusSet](#)
- [ODSession::Tokenize](#)

-----

## RelinquishFocus - Topics

### Class:

ODArbitrator

Select an item:

[Syntax](#)

[Parameters](#)

---

## RelinquishFocusSet

---

### RelinquishFocusSet - Syntax

This method is called by the current owner of the each focus in the specified focus set to relinquish ownership of it.

```
#define INCL_ODARBITRATOR
#define INCL_ODAPI
#include <os2.h>

ODFocusSet      *focusSet;
ODFrame         *relinquishingFrame;

RelinquishFocusSet(focusSet, relinquishingFrame);
```

---

### RelinquishFocusSet Parameter - focusSet

**focusSet** (ODFocusSet \*) - input  
A reference to a focus set containing the foci to be relinquished.

---

### RelinquishFocusSet Parameter - relinquishingFrame

**relinquishingFrame** (ODFrame \*) - input  
A reference to the frame relinquishing ownership of each focus in the specified focus set.

---

### RelinquishFocusSet - Return Value

None.

---

### RelinquishFocusSet - Parameters

**focusSet** (ODFocusSet \*) - input

A reference to a focus set containing the foci to be relinquished.

**relinquishingFrame** (ODFrame \*) - input

A reference to the frame relinquishing ownership of each focus in the specified focus set.

None.

---

## RelinquishFocusSet - Exception Handling

kODErrFocusNotRegistered

One of the specified foci is not registered.

---

## RelinquishFocusSet - Related Methods

### Related Methods

- [ODArbitrator::RelinquishFocus](#)

---

## RelinquishFocusSet - Topics

### Class:

ODArbitrator

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Exception Handling](#)

[Related Methods](#)

---

## RequestFocus

---

## RequestFocus - Syntax

This method requests that the ownership of the specified focus be assigned to the specified frame.

```
#define INCL_ODARBITRATOR
```



```

#define INCL_ODAPI
#include <os2.h>

ODTypeToken    focus;
ODFrame        *requestingFrame;
ODBoolean      rc;

rc = RequestFocus(focus, requestingFrame);

```

## RequestFocus Parameter - focus

**focus** (ODTypeToken) - input

A tokenized string representing the focus type whose ownership is being requested.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus	The frame with the clipboard focus has access to the clipboard.
kODKeyFocus	The frame with keyboard focus receives keyboard events (excluding page-movement key events).
kODMenuFocus	The frame with menu focus receives menu events.
kODModalFocus	A frame with modal focus is notifying other frames that it is the only currently modal frame.
kODMouseFocus	The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.
kODScrollingFocus	The frame with the scrolling focus receives page-movement key events (such as <b>PageUp</b> ). This frame does not need to own keyboard focus.
kODSelectionFocus	The frame with the selection focus receives modified mouse-click events ( <b>Shift</b> +click and <b>Ctrl</b> +click). OpenDoc draws the active frame border around all facets of this frame.

## RequestFocus Parameter - requestingFrame

**requestingFrame** (ODFrame \*) - input

A reference to the frame requesting the focus.

## RequestFocus Return Value - rc

**rc** (ODBoolean) - returns

A flag indicating whether the frame obtained the focus.

kODTrue	The frame obtained the focus.
kODFalse	The frame did not obtain the focus.

## RequestFocus - Parameters

**focus** ([ODTypeToken](#)) - input

A tokenized string representing the focus type whose ownership is being requested.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

**kODClipboardFocus**

The frame with the clipboard focus has access to the clipboard.

**kODKeyFocus**

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

**kODMenuFocus**

The frame with menu focus receives menu events.

**kODModalFocus**

A frame with modal focus is notifying other frames that it is the only currently modal frame.

**kODMouseFocus**

The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

**kODScrollingFocus**

The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

**kODSelectionFocus**

The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

**requestingFrame** ([ODFrame \\*](#)) - input

A reference to the frame requesting the focus.

**rc** ([ODBoolean](#)) - returns

A flag indicating whether the frame obtained the focus.

**kODTrue**

The frame obtained the focus.

**kODFalse**

The frame did not obtain the focus.

-----

## RequestFocus - Remarks

Your part calls this method to request a single focus for one of its frames; to request multiple foci, your part calls the arbitrator's [RequestFocusSet](#) method. This return value indicates whether the specified frame obtained ownership of the specified focus.

If the specified focus is nonexclusive, the specified frame is automatically granted ownership of the focus. If it is exclusive, the focus module calls the [BeginRelinquishFocus](#) method of the current owner's part to see if the current owner is willing to give it up.

If the request is granted, the new ownership relationship is stored in the relevant focus modules. If the request fails, the existing ownership relationships remain intact.

If the request is granted, the arbitrator contains a reference to the frame. Parts should relinquish the focus in the [DisplayFrameClosed](#) or [DisplayFrameRemoved](#) methods.

-----

## RequestFocus - Exception Handling

**kODErrFocusNotRegistered**

The requested focus is not registered.

-----

## RequestFocus - Related Methods

**Related Methods**

- [ODArbitrator::RequestFocusSet](#)
- [ODPart::BeginRelinquishFocus](#)
- [ODSession::Tokenize](#)

-----

## RequestFocus - Topics

### Class:

[ODArbitrator](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

-----

## RequestFocusSet

-----

## RequestFocusSet - Syntax

This method requests that the ownership of each focus in the specified focus set be assigned to the specified frame.

```
#define INCL_ODARBITRATOR
#define INCL_ODAPI
#include <os2.h>

ODFocusSet      *focusSet;
ODFrame          *requestingFrame;
ODBoolean        rc;

rc = RequestFocusSet(focusSet, requestingFrame);
```

-----

## RequestFocusSet Parameter - focusSet

**focusSet** (ODFocusSet \*) - input

A reference to a focus set being requested.

-----

## RequestFocusSet Parameter - requestingFrame

**requestingFrame** (ODFrame \*) - input  
A reference to a frame requesting the focus.

---

## RequestFocusSet Return Value - rc

**rc** (ODBoolean) - returns  
A flag indicating whether the frame obtained the focus set.

kODTrue	The frame obtained the focus set.
kODFalse	The frame did not obtain the focus set.

---

## RequestFocusSet - Parameters

**focusSet** (ODFocusSet \*) - input  
A reference to a focus set being requested.

**requestingFrame** (ODFrame \*) - input  
A reference to a frame requesting the focus.

**rc** (ODBoolean) - returns  
A flag indicating whether the frame obtained the focus set.

kODTrue	The frame obtained the focus set.
kODFalse	The frame did not obtain the focus set.

---

## RequestFocusSet - Remarks

For the requested ownership to be granted, all existing owners of exclusive foci within the specified set must be willing to relinquish these same foci. If all of the foci, exclusive and nonexclusive, are attainable, then the request is granted; however, even if one focus is unattainable, ownership of the focus set is not granted.

If the request is granted, the new ownership relationships are stored in the relevant focus modules. If the request fails, the existing ownership relationships remains intact.

If the request is granted, the arbitrator contains a reference to the frame. Parts should relinquish the focus in the [DisplayFrameClosed](#) or [DisplayFrameRemoved](#) methods.

---

## RequestFocusSet - Exception Handling

kODErrFocusNotRegistered	One of the requested foci is not registered.
--------------------------	--

---

## RequestFocusSet - Related Methods

## Related Methods

- [ODArbitrator::RequestFocus](#)

---

# RequestFocusSet - Topics

## Class:

ODArbitrator

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

# TransferFocus

---

## TransferFocus - Syntax

This method directly transfers a focus from its current owner to another.

```
#define INCL_ODARBITRATOR
#define INCL_ODAPI
#include <os2.h>

ODTypeToken    focus;
ODFrame        *transferringFrame;
ODFrame        *newOwner;

TransferFocus(focus, transferringFrame, newOwner);
```

---

## TransferFocus Parameter - focus

**focus** ([ODTypeToken](#)) - input

A tokenized string representing the focus type to be transferred.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the clipboard focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus	The frame with menu focus receives menu events.
kODModalFocus	A frame with modal focus is notifying other frames that it is the only currently modal frame.
kODMouseFocus	The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.
kODScrollingFocus	The frame with the scrolling focus receives page-movement key events (such as <b>PageUp</b> ). This frame does not need to own keyboard focus.
kODSelectionFocus	The frame with the selection focus receives modified mouse-click events ( <b>Shift</b> +click and <b>Ctrl</b> +click). OpenDoc draws the active frame border around all facets of this frame.

-----

## TransferFocus Parameter - transferringFrame

**transferringFrame** (ODFrame \*) - input  
A reference to a frame transferring ownership of the focus. This frame does not need to be the current owner.

-----

## TransferFocus Parameter - newOwner

**newOwner** (ODFrame \*) - input  
A reference to a frame that is to be the new focus owner.

-----

## TransferFocus - Return Value

None.

-----

## TransferFocus - Parameters

**focus** (ODTypeToken) - input  
A tokenized string representing the focus type to be transferred.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus	The frame with the <a href="#">clipboard</a> focus has access to the clipboard.
kODKeyFocus	The frame with keyboard focus receives keyboard events (excluding page-movement key events).
kODMenuFocus	The frame with menu focus receives menu events.
kODModalFocus	A frame with modal focus is notifying other frames that it is the only currently modal frame.
kODMouseFocus	The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is

located.

kODScrollingFocus

The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

kODSelectionFocus

The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

**transferringFrame** (ODFrame \*) - input

A reference to a frame transferring ownership of the focus. This frame does not need to be the current owner.

**newOwner** (ODFrame \*) - input

A reference to a frame that is to be the new focus owner.

None.

-----

## TransferFocus - Remarks

Your part calls this method, instead of methods that relinquish a focus, to restore ownership to a previous owner. For example, when a part requests the modal focus, as it does when displaying a modal dialog box, that part should save the previous owner of the modal focus and transfer ownership back to the previous owner when the dialog box is dismissed. This technique may be necessary if modal dialogs can be nested.

This method calls the new owner's [FocusAcquired](#) method if the new owner is not the transferring frame. If the previous owner is not the transferring frame, OpenDoc also calls the previous owner's [FocusLost](#) method.

-----

## TransferFocus - Related Methods

### Related Methods

- [ODArbitrator::TransferFocusSet](#)
- [ODPart::FocusAcquired](#)
- [ODPart::FocusLost](#)
- [ODSession::Tokenize](#)

-----

## TransferFocus - Topics

### Class:

ODArbitrator

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

-----

## TransferFocusSet

---

## TransferFocusSet - Syntax

This method assigns the specified frame as the owner of each focus in the specified focus set.

```
#define INCL_ODARBITRATOR
#define INCL_ODAPI
#include <os2.h>

ODFocusSet    *focusSet;
ODFrame       *transferringFrame;
ODFrame       *newOwner;

TransferFocusSet(focusSet, transferringFrame,
                newOwner);
```

---

## TransferFocusSet Parameter - focusSet

**focusSet** (ODFocusSet \*) - input  
A reference to a focus set to be transferred.

---

## TransferFocusSet Parameter - transferringFrame

**transferringFrame** (ODFrame \*) - input  
A reference to a frame transferring ownership of the focus set.

---

## TransferFocusSet Parameter - newOwner

**newOwner** (ODFrame \*) - input  
A reference to a frame that is to be the new focus set owner.

---

## TransferFocusSet - Return Value

None.

---

## TransferFocusSet - Parameters



**focusSet** (ODFocusSet \*) - input  
A reference to a focus set to be transferred.

**transferringFrame** (ODFrame \*) - input  
A reference to a frame transferring ownership of the focus set.

**newOwner** (ODFrame \*) - input  
A reference to a frame that is to be the new focus set owner.

None.

---

## TransferFocusSet - Remarks

This method calls the new owner's [FocusAcquired](#) method if the new owner is not the transferring frame. If the previous owner is not the transferring frame, OpenDoc also calls the previous owner's [FocusLost](#) method.

---

## TransferFocusSet - Related Methods

### Related Methods

- [ODArbitrator::TransferFocus](#)
- [ODPart::FocusAcquired](#)
- [ODPart::FocusLost](#)

---

## TransferFocusSet - Topics

### Class:

ODArbitrator

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## UnregisterFocus

---

## UnregisterFocus - Syntax

This method removes the association between the specified focus and the focus module that manages it.

```
#define INCL_ODARBITRATOR
#define INCL_ODAPI
#include <os2.h>

ODTypeToken    focus;

UnregisterFocus(focus);
```

## UnregisterFocus Parameter - focus

**focus** (ODTypeToken) - input

A tokenized string representing the focus type to be removed.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the clipboard focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus

The frame with menu focus receives menu events.

kODModalFocus

A frame with modal focus is notifying other frames that it is the only currently modal frame.

kODMouseFocus

The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

kODScrollingFocus

The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

kODSelectionFocus

The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

## UnregisterFocus - Return Value

None.

## UnregisterFocus - Parameters

**focus** (ODTypeToken) - input

A tokenized string representing the focus type to be removed.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the clipboard focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus

	The frame with menu focus receives menu events.
kODModalFocus	A frame with modal focus is notifying other frames that it is the only currently modal frame.
kODMouseFocus	The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.
kODScrollingFocus	The frame with the scrolling focus receives page-movement key events (such as <b>PageUp</b> ). This frame does not need to own keyboard focus.
kODSelectionFocus	The frame with the selection focus receives modified mouse-click events ( <b>Shift</b> +click and <b>Ctrl</b> +click). OpenDoc draws the active frame border around all facets of this frame.
None.	

-----

## UnregisterFocus - Exception Handling

kODErrFocusNotRegistered	The requested focus is not registered.
--------------------------	--

-----

## UnregisterFocus - Related Methods

### Related Methods

- [ODSession::Tokenize](#)

-----

## UnregisterFocus - Topics

### Class:

ODArbitrator

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Exception Handling](#)  
[Related Methods](#)

-----

## ODBinding

**Class Definition File:** ODBINDNG.IDL

### Class Hierarchy

SOMObject  
   ODObject  
     **ODBinding**

### Description

An object of the ODBinding class represents the OpenDoc binding object that performs the run-time binding of part editors to the parts in a document.

When a document is opened, the session object creates a single binding object. All parts of that document share the binding object; the document shell or a container application can obtain a reference to it by calling the session object's [GetBinding](#) method.

OpenDoc binds part editors to part data when a part is read in or when its part editor is changed. The [binding](#) object gathers information provided by the installed part editors, preferences specified by the user, and part-kind information stored with parts. It uses that information to choose an editor for each part in the document.

For more information about binding, see the chapter on OpenDoc run-time features in the *OpenDoc Programming Guide* .

## Methods

The methods defined by the ODBinding class include:

- [ChooseEditorForPart](#)
- [GetContainerSuite](#)

## Overridden Methods

There are currently no methods overridden by the ODBinding class.

---

# ChooseEditorForPart

---

## ChooseEditorForPart - Syntax

This method chooses the editor to be used to edit the specified part.

```
#define INCL_ODBINDING
#define INCL_ODAPI
#include <os2.h>

ODStorageUnit    *thePartSU;
ODType           newKind;
ODEditor         rv;

rv = ChooseEditorForPart (thePartSU, newKind);
```

---

## ChooseEditorForPart Parameter - thePartSU

**thePartSU** (ODStorageUnit \*) - input  
A reference to the part's storage unit.

---

## ChooseEditorForPart Parameter - newKind

**newKind** (ODType) - input  
The part kind of the new part being created or KODNULL if the specified part already exists.

---

## ChooseEditorForPart Return Value - rv

**rv** ([ODEditor](#)) - returns  
An opaque, platform-specific value specifying the chosen part editor.

---

## ChooseEditorForPart - Parameters

**thePartSU** ([ODStorageUnit](#) \*) - input  
A reference to the part's storage unit.

**newKind** ([ODType](#)) - input  
The part kind of the new part being created or `KODNULL` if the specified part already exists.

**rv** ([ODEditor](#)) - returns  
An opaque, platform-specific value specifying the chosen part editor.

---

## ChooseEditorForPart - Remarks

This method is called by the container suite. The document shell, container applications, and parts cannot call this method.

This method chooses the appropriate editor for the specified part based on the editors that are installed on the machine and the part kind of the specified part. For a new part that is being created, the specified storage unit is empty and the *newKind* parameter specifies the part kind. For a part that exists, the storage unit contains the part kinds.

---

## ChooseEditorForPart - Topics

**Class:**  
[ODBinding](#)

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

## GetContainerSuite

---

## GetContainerSuite - Syntax

This method returns a container suite ID to be used for instantiating a container suite by name.

```
#define INCL_ODBINDING
#define INCL_ODAPI
#include <os2.h>

ODContainerType    containerType;
ODContainerSuite   rv;

rv = GetContainerSuite(containerType);
```

-----

## GetContainerSuite Parameter - containerType

**containerType** ([ODContainerType](#)) - input

The type of the container object. This parameter can be set to one of the following values:

kODBentoFileContainer

The Bento container class for documents.

kODBentoMemoryContainer

The Bento container class for documents on this platform.

kODDefaultFileContainer

The default container type for documents on this platform.

kODDefaultMemoryContainer

The default container type for drag-and-drop operations and the clipboard on this platform.

-----

## GetContainerSuite Return Value - rv

**rv** ([ODContainerSuite](#)) - returns

The container suite ID identifying the specified container suite.

-----

## GetContainerSuite - Parameters

**containerType** ([ODContainerType](#)) - input

The type of the container object. This parameter can be set to one of the following values:

kODBentoFileContainer

The Bento container class for documents.

kODBentoMemoryContainer

The Bento container class for documents on this platform.

kODDefaultFileContainer

The default container type for documents on this platform.

kODDefaultMemoryContainer

The default container type for drag-and-drop operations and the clipboard on this platform.

**rv** ([ODContainerSuite](#)) - returns

The container suite ID identifying the specified container suite.

---

# GetContainerSuite - Topics

## Class:

ODBinding

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

---

# ODCanvas

**Class Definition File:** CANVAS.IDL

## Class Hierarchy

SOMObject

ODObject

ODBaseCanvas

**ODCanvas**

## Description

An object of the ODCanvas class is a wrapper for a graphics-system-specific drawing structure that represents a *drawing environment* -the environment for constructing an image.

A canvas object represents a drawing environment (the destination for drawing calls) for any of the available graphics systems that your part uses. Your part uses the standard platform drawing calls for your graphics system to render its content on a canvas. A *canvas* can refer to anything that a graphics system knows how to draw into; for example, a graphics port, print job, offscreen pixel buffer, bit map, structured display list, or stream of PostScript code. A graphic-system-specific drawing structure may retain state information (such as pen color) that influences how the drawing calls are interpreted.

Your part creates a canvas object by calling the window-state object's [CreateCanvas](#) method. To create a canvas to attach to a particular facet, your part can call that facet's [CreateCanvas](#) method.

Each canvas object includes one or more graphic-system-specific drawing structures, which are not deleted when the canvas is released. If you create a canvas, you must separately create the underlying drawing structure, and you are responsible for deleting that drawing structure when the canvas is released.

## Canvas Characteristics

A canvas must be either dynamic or static:

dynamic canvas

A drawing canvas that is interactive. Windows which can be scrolled or paged to display different portions of a part's content are good examples of dynamic canvases.

static canvas

A drawing canvas that cannot be changed once it is rendered; for instance, one that cannot be scrolled. A printed image (or onscreen print preview) is an example of a static canvas because the user cannot scroll elements on the page or otherwise interact with it, once it is drawn.

Your part can displays parts differently based on this distinction. For example, you might use scroll bars on the screen but not on printed material. You might also use different drawing calls for printing than for screen display because printing varies from platform to platform.

A canvas is also defined as being either onscreen or offscreen:

onscreen canvas

The main canvas of the window or print job.

offscreen canvas

This canvas is used to improve performance by allowing you to draw a complex image to an offscreen cache, and then quickly transfer the completed image to the onscreen canvas, with full freedom to alter or distort the image in the process. For instance, you can create an offscreen canvas to do double-buffering or image manipulation, such as changing the tinting or translucency of an image.

When a part creates a canvas, it specifies, for the lifetime of the canvas, whether the canvas is dynamic or static and whether it is onscreen or offscreen.

A canvas is further defined in terms of a transformation matrix and its coordinate bias:

bias transform

A transform matrix that describes the transform that is applied to measurements in a canvas's coordinate space to change them into standard platform-normal coordinates.

Bias transforms are used to negotiate between the containing part and the embedded part. The negotiation occurs when several canvases, each defined in its own coordinate space, are combined into a single coordinate system. Facets use bias transforms to convert geometry from one coordinate space to the other, usually so that a part can use the geometry to display on its canvas. Each canvas is scaled, rotated, and translated by redefinition of its coordinates in the bias canvas's coordinate space. Thus, once you set up your offscreen canvas for drawing in your own coordinate system, you can also use it to make sure that all point, frame, and facet geometry is properly converted for you.

coordinate bias,

This is defined by the bias transform. It is the difference between the canvas's coordinate space and the standard platform-normal coordinate space. The coordinate bias usually takes the form of an offset in the origin, a change in the polarity of one or more axes, and possibly a change in scale.

### Offscreen Imaging

Canvases can be attached to individual facets. If a particular facet in a window's facet hierarchy has an attached canvas, it and all of its embedded facets (and their embedded facets, and so on) draw to that canvas. Each facet inherits its canvas from its containing facet; the inherited canvas is called the *parent canvas*. For most drawing, only a window's root facet needs a canvas.

If a particular part needs an offscreen canvas, however, it can attach a canvas to one of its facets on a display frame. The canvas has a reference to that facet and also a reference to the part that created the canvas and attached it to a facet—the *owning part*. The reference enables the canvas to notify the owning part that its content has changed and that it needs to be updated. The owning part is responsible for copying the image of the offscreen canvas to its parent canvas during updates.

The owning part of a canvas does not need to be the same as the facet's part. For instance, a containing part may customize the drawing of an embedded part by assigning the facet of the embedded part to an offscreen canvas. In this case, the containing part must make itself the owning part so that it can control the drawing of the facet on the screen.

For added convenience, offscreen canvases maintain updating information that mirrors their onscreen equivalents. This lets embedded parts interact with their drawing environment in a consistent manner, whether the parts are displayed in the window canvas or in an offscreen canvas.

### **Methods**

The methods defined by the ODCanvas class include:

- [AcquireBiasTransform](#)
- [AcquireOwner](#)
- [AcquireUpdateShape](#)
- [GetFacet](#)
- [GetPlatformCanvas](#)
- [GetPlatformPrintJob](#)
- [HasPlatformCanvas](#)
- [HasPlatformPrintJob](#)
- [Invalidate](#)
- [IsDynamic](#)
- [IsOffscreen](#)
- [ResetUpdateShape](#)
- [SetBiasTransform](#)
- [SetFacet](#)
- [SetOwner](#)
- [SetPlatformCanvas](#)
- [SetPlatformPrintJob](#)
- [Validate](#)

### **Overridden Methods**

There are currently no methods overridden by the ODCanvas class.

---

## AcquireBiasTransform



---

# AcquireBiasTransform - Syntax

This method returns a reference to the bias transform associated with this canvas.

```
#define INCL_ODCANVAS
#define INCL_ODAPI
#include <os2.h>

ODTransform      *rv;

rv = AcquireBiasTransform();
```

---

## AcquireBiasTransform Return Value - rv

**rv** (ODTransform \*) - returns

A reference to the bias transform associated with this canvas or KODNULL if no bias transform has previously been assigned to this canvas.

---

## AcquireBiasTransform - Parameters

**rv** (ODTransform \*) - returns

A reference to the bias transform associated with this canvas or KODNULL if no bias transform has previously been assigned to this canvas.

---

## AcquireBiasTransform - Remarks

If you call methods that modify the returned transform, you must then call this canvas's [SetBiasTransform](#) method to set its bias transform to the modified transform.

This method increments the reference count of the returned transform object. When you have finished using that transform object, you should call its [Release](#) method.

---

## AcquireBiasTransform - Related Methods

### Related Methods

- [ODCanvas::SetBiasTransform](#)

---

## AcquireBiasTransform - Topics

**Class:**

ODCanvas

Select an item:

[Syntax](#)[Parameters](#)[Returns](#)[Remarks](#)[Related Methods](#)

---

## AcquireOwner

---

### AcquireOwner - Syntax

This method returns a reference to the part that owns this canvas.

```
#define INCL_ODCANVAS
#define INCL_ODAPI
#include <os2.h>
```

```
ODPart      *rv;
```

```
rv = AcquireOwner();
```

---

### AcquireOwner Return Value - rv

**rv** (ODPart \*) - returns

A reference to the part that owns this canvas or kODNULL if the part does not exist.

---

### AcquireOwner - Parameters

**rv** (ODPart \*) - returns

A reference to the part that owns this canvas or kODNULL if the part does not exist.

---

### AcquireOwner - Remarks

This method increments the reference count of the returned part object. When you have finished using that part object, you should call its [Release](#) method.

---

# AcquireOwner - Related Methods

## Related Methods

- [ODCanvas::SetOwner](#)

---

# AcquireOwner - Topics

## Class:

ODCanvas

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

# AcquireUpdateShape

---

# AcquireUpdateShape - Syntax

This method returns a reference to the shape object defining the area of this canvas that needs to be updated.

```
#define INCL_ODCANVAS
#define INCL_ODAPI
#include <os2.h>

ODShape      *rv;

rv = AcquireUpdateShape();
```

---

# AcquireUpdateShape Return Value - rv

**rv** (ODShape \*) - returns

A reference to the shape object defining the area of this canvas that needs to be updated.

---

# AcquireUpdateShape - Parameters

**rv** (ODShape \*) - returns  
A reference to the shape object defining the area of this canvas that needs to be updated.

---

## AcquireUpdateShape - Remarks

OpenDoc calls this method internally. Your part modifies the update shape of a canvas by calling its facet's [Invalidate](#) and [Validate](#) methods.

This method increments the reference count of the returned shape object. When the caller has finished using that shape object, it should call the shape's [Release](#) method.

---

## AcquireUpdateShape - Related Methods

### Related Methods

- [ODCanvas::Invalidate](#)
  - [ODCanvas::ResetUpdateShape](#)
  - [ODCanvas::Validate](#)
  - [ODFacet::Invalidate](#)
  - [ODFacet::Validate](#)
- 

## AcquireUpdateShape - Topics

### Class:

[ODCanvas](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## GetFacet

---

## GetFacet - Syntax

This method returns a reference to the facet associated with this canvas.

```
#define INCL_ODCANVAS  
#define INCL_ODAPI
```

```
#include <os2.h>

ODFacet      *rv;

rv = GetFacet();
```

---

## GetFacet Return Value - rv

**rv** (ODFacet \*) - returns  
A reference to the facet associated with this canvas.

---

## GetFacet - Parameters

**rv** (ODFacet \*) - returns  
A reference to the facet associated with this canvas.

---

## GetFacet - Related Methods

### Related Methods

- [ODCanvas::SetFacet](#)
- 

## GetFacet - Topics

### Class:

ODCanvas

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Related Methods](#)

---

## GetPlatformCanvas

---

## GetPlatformCanvas - Syntax

This method returns the drawing structure for the specified graphics system for this canvas.

```
#define INCL_ODCANVAS
#define INCL_ODAPI
#include <os2.h>

ODGraphicsSystem      g;
ODPlatformCanvas      *rv;

rv = GetPlatformCanvas(g);
```

---

## GetPlatformCanvas Parameter - g

**g** ([ODGraphicsSystem](#)) - input

The graphics system you want to use for this canvas. Valid graphics systems are platform dependant. For OS/2, this parameter should always be set to [KODGPI](#).

---

## GetPlatformCanvas Return Value - rv

**rv** ([ODPlatformCanvas](#) \*) - returns

A pointer to the graphics-system specific drawing structure for this canvas or [KODNULL](#) if this canvas does not have and cannot generate a drawing structure for the specified graphics system. Valid values for this structure are graphics-system specific.

On OS/2, [ODPlatformCanvas](#) is a SOM class which encapsulates one or more GPI presentation spaces.

---

## GetPlatformCanvas - Parameters

**g** ([ODGraphicsSystem](#)) - input

The graphics system you want to use for this canvas. Valid graphics systems are platform dependant. For OS/2, this parameter should always be set to [KODGPI](#).

**rv** ([ODPlatformCanvas](#) \*) - returns

A pointer to the graphics-system specific drawing structure for this canvas or [KODNULL](#) if this canvas does not have and cannot generate a drawing structure for the specified graphics system. Valid values for this structure are graphics-system specific.

On OS/2, [ODPlatformCanvas](#) is a SOM class which encapsulates one or more GPI presentation spaces.

---

## GetPlatformCanvas - Remarks

You call this method to get the graphics-system-specific drawing structure (for instance, a window or view port) when you need to draw into a facet.

On OS/2, this method returns a reference to a platform canvas object which encapsulates one or more GPI presentation space. Your part calls this method when it needs to obtain a presentation space for drawing. You obtain the presentation space from the [ODPlatformCanvas](#) object by calling its [GetPS](#) method, specifying the facet that you want to draw in. See [ODPlatformCanvas](#) for more information.

You must specify the graphics system because some implementations allow you to set drawing structures for two or more graphics systems simultaneously. The specific graphics system supported is implementation dependent.

For more information on graphic-system-specific drawing structures, see the *Graphics Programming Interface Programming Guide* .

---

## GetPlatformCanvas - Exception Handling

kODErrInvalidGraphicsSystem

This implementation of OpenDoc does not support the specified graphics system, that graphics system is not installed or available, or this canvas has no drawing structure for that graphics system.

---

## GetPlatformCanvas - Related Methods

### Related Methods

- [ODCanvas::HasPlatformCanvas](#)
- [ODCanvas::SetPlatformCanvas](#)

---

## GetPlatformCanvas - Topics

### Class:

[ODCanvas](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

## GetPlatformPrintJob

---

## GetPlatformPrintJob - Syntax

This method returns the print job for a specified graphics system for this canvas.

```
#define INCL_ODCANVAS
#define INCL_ODAPI
#include <os2.h>
```

```

ODGraphicsSystem    g;
ODPlatformPrintJob  rv;

rv = GetPlatformPrintJob(g);

```

---

## GetPlatformPrintJob Parameter - g

**g** ([ODGraphicsSystem](#)) - input

The graphics system you want to use for this canvas. Valid graphics systems are platform dependent.

For OS/2, this parameter should always be set to kODGPI.

---

## GetPlatformPrintJob Return Value - rv

**rv** ([ODPlatformPrintJob](#)) - returns

The graphics-system-specific print job for this canvas. You must cast the return value to a valid graphics-system type before using it.

On OS/2, this type is a pointer to a [PRINTDEST](#) structure. The caller should not modify or free the memory in this structure.

---

## GetPlatformPrintJob - Parameters

**g** ([ODGraphicsSystem](#)) - input

The graphics system you want to use for this canvas. Valid graphics systems are platform dependent.

For OS/2, this parameter should always be set to kODGPI.

**rv** ([ODPlatformPrintJob](#)) - returns

The graphics-system-specific print job for this canvas. You must cast the return value to a valid graphics-system type before using it.

On OS/2, this type is a pointer to a [PRINTDEST](#) structure. The caller should not modify or free the memory in this structure.

---

## GetPlatformPrintJob - Remarks

If this canvas has a print job for the specified graphics system, this method returns that print job. You need to call this method only when you are creating a static canvas to be used as a print job.

A canvas can have only one print job (for one graphics system) even if it has drawing structures for more than one graphics system.

---

## GetPlatformPrintJob - Exception Handling

kODErrInvalidGraphicsSystem

The implementation of OpenDoc does not support the specified



---

## GetPlatformPrintJob - Related Methods

### Related Methods

- [ODCanvas::HasPlatformPrintJob](#)
- [ODCanvas::SetPlatformPrintJob](#)

---

## GetPlatformPrintJob - Topics

### Class:

[ODCanvas](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

## HasPlatformCanvas

---

## HasPlatformCanvas - Syntax

This method indicates whether this canvas has or can generate drawing structure for the specified graphics system.

```
#define INCL_ODCANVAS
#define INCL_ODAPI
#include <os2.h>

ODGraphicsSystem    g;
ODBoolean           rv;

rv = HasPlatformCanvas(g);
```

---

## HasPlatformCanvas Parameter - g

**g** ([ODGraphicsSystem](#)) - input

The graphics system you want to check for this canvas. Valid graphics systems are platform-dependent.

For OS/2, this parameter should always be set to kODGPI.

-----

## HasPlatformCanvas Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating whether this canvas has or can generate a drawing structure for the specified graphics system. This parameter can be set to one of the following values:

kODTrue

This canvas has or can generate a drawing structure for the specified graphics system.

kODFalse

This canvas does not have and cannot generate a graphics-system-specific drawing structure.

-----

## HasPlatformCanvas - Parameters

**g** ([ODGraphicsSystem](#)) - input

The graphics system you want to check for this canvas. Valid graphics systems are platform-dependent.

For OS/2, this parameter should always be set to kODGPI.

**rv** ([ODBoolean](#)) - returns

A flag indicating whether this canvas has or can generate a drawing structure for the specified graphics system. This parameter can be set to one of the following values:

kODTrue

This canvas has or can generate a drawing structure for the specified graphics system.

kODFalse

This canvas does not have and cannot generate a graphics-system-specific drawing structure.

-----

## HasPlatformCanvas - Related Methods

### Related Methods

- [ODCanvas::GetPlatformCanvas](#)
- [ODCanvas::SetPlatformCanvas](#)

-----

## HasPlatformCanvas - Topics

### Class:

ODCanvas

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Related Methods](#)

---

# HasPlatformPrintJob

---

## HasPlatformPrintJob - Syntax

This method indicates whether this canvas has a print job for the specified graphics system.

```
#define INCL_ODCANVAS
#define INCL_ODAPI
#include <os2.h>

ODGraphicsSystem    g;
ODBoolean           rv;

rv = HasPlatformPrintJob(g);
```

---

## HasPlatformPrintJob Parameter - g

**g** ([ODGraphicsSystem](#)) - input

The graphics system you want to check for this canvas. Valid graphics systems are platform-dependent.

For OS/2, this parameter should always be set to kODGPI.

---

## HasPlatformPrintJob Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating whether this canvas has a print job for the specified graphics system. This parameter can be set to one of the following values:

kODTrue

This canvas has a print job for the specified graphics system.

kODFalse

This canvas does not have a print job for the specified graphics system.

---

## HasPlatformPrintJob - Parameters

**g** ([ODGraphicsSystem](#)) - input

The graphics system you want to check for this canvas. Valid graphics systems are platform-dependent.

For OS/2, this parameter should always be set to kODGPI.

**rv** ([ODBoolean](#)) - returns

A flag indicating whether this canvas has a print job for the specified graphics system. This parameter can be set to one of the following values:

kODTrue

This canvas has a print job for the specified graphics system.

kODFalse

This canvas does not have a print job for the specified graphics system.

-----

## HasPlatformPrintJob - Related Methods

### Related Methods

- [ODCanvas::GetPlatformPrintJob](#)
- [ODCanvas::SetPlatformPrintJob](#)

-----

## HasPlatformPrintJob - Topics

### Class:

[ODCanvas](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Related Methods](#)

-----

## Invalidate

-----

## Invalidate - Syntax

This method adds the specified area to the update shape for this canvas, ensuring that the specified area for this canvas is updated.

```
#define INCL_ODCANVAS
#define INCL_ODAPI
#include <os2.h>
```

```
ODShape      *shape;
```

```
Invalidate(shape);
```

-----

## Invalidate Parameter - shape

**shape** (ODShape \*) - input

A reference to the shape object defining the area to be added to the update shape for this canvas.

---

## Invalidate - Return Value

None.

---

## Invalidate - Parameters

**shape** (ODShape \*) - input

A reference to the shape object defining the area to be added to the update shape for this canvas.

None.

---

## Invalidate - Remarks

OpenDoc calls this method internally to mark the area of a canvas that needs updating. Your part typically calls its facet's [Invalidate](#) method instead of this method because that method transforms and clips the shape from the coordinate space of the facet to the coordinate space of its canvas.

---

## Invalidate - Related Methods

### Related Methods

- [ODCanvas::Validate](#)
  - [ODFacet::Invalidate](#)
  - [ODFrame::Invalidate](#)
- 

## Invalidate - Topics

### Class:

ODCanvas

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

# IsDynamic

---

## IsDynamic - Syntax

This method indicates whether this canvas is dynamic.

```
#define INCL_ODCANVAS
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    rv;

rv = IsDynamic();
```

---

## IsDynamic Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the canvas is dynamic. This parameter can be set to one of the following values:

kODTrue	This canvas is dynamic.
kODFalse	This canvas is static.

---

## IsDynamic - Parameters

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the canvas is dynamic. This parameter can be set to one of the following values:

kODTrue	This canvas is dynamic.
kODFalse	This canvas is static.

---

## IsDynamic - Remarks

The dynamic or static characteristic of a canvas is set when the canvas is created and cannot be changed.

---

# IsDynamic - Topics

## Class:

ODCanvas

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

---

## IsOffscreen

---

## IsOffscreen - Syntax

This method indicates whether this canvas is offscreen.

```
#define INCL_ODCANVAS
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    rv;

rv = IsOffscreen();
```

---

## IsOffscreen Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the canvas is offscreen. This parameter can be set to one of the following values:

kODTrue	This canvas is offscreen.
kODFalse	This canvas is onscreen.

---

## IsOffscreen - Parameters

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the canvas is offscreen. This parameter can be set to one of the following values:

kODTrue	This canvas is offscreen.
kODFalse	

This canvas is onscreen.

---

## IsOffscreen - Remarks

The onscreen or offscreen characteristic of a canvas is set when the canvas is created and cannot be changed.

---

## IsOffscreen - Topics

### Class:

ODCanvas

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

---

## ResetUpdateShape

---

## ResetUpdateShape - Syntax

This method sets the update shape for this canvas to the empty shape.

```
#define INCL_ODCANVAS
#define INCL_ODAPI
#include <os2.h>
```

```
ResetUpdateShape();
```

---

## ResetUpdateShape - Return Value

None.

---

## ResetUpdateShape - Parameters



None.

---

## ResetUpdateShape - Remarks

OpenDoc calls this method internally while processing update events and after all the invalidated canvases have been updated.

---

## ResetUpdateShape - Related Methods

### Related Methods

- [ODCanvas::AcquireUpdateShape](#)
- 

## ResetUpdateShape - Topics

### Class:

ODCanvas

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## SetBiasTransform

---

## SetBiasTransform - Syntax

This method assigns the specified bias transform to this canvas.

```
#define INCL_ODCANVAS
#define INCL_ODAPI
#include <os2.h>
```

```
ODTransform    *x;

SetBiasTransform(x);
```

---

# SetBiasTransform Parameter - x

**x** (ODTransform \*) - input

A reference to the bias transform to be assigned to this canvas.

---

## SetBiasTransform - Return Value

None.

---

## SetBiasTransform - Parameters

**x** (ODTransform \*) - input

A reference to the bias transform to be assigned to this canvas.

None.

---

## SetBiasTransform - Remarks

This bias transform is calculated by concatenating the internal transform (if any) of this canvas's owning frame with the internal transform of the root frame. You can use this method to add a vertical flip and offset between frames of graphics systems whose coordinate spaces have different handedness, for example, Mac OS- and Windows-based frames on an OS/2 system.

After this method executes successfully, any preexisting bias transform is released and this canvas owns the new transform. You can call the [AcquireBiasTransform](#) method of this canvas to obtain a reference to the resulting bias transform.

You should release the transform object **x** immediately after passing it as a parameter to this method, without using or modifying it further.

---

## SetBiasTransform - Related Methods

### Related Methods

- [ODCanvas::AcquireBiasTransform](#)
- 

## SetBiasTransform - Topics

**Class:**

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

## SetFacet

---

### SetFacet - Syntax

This method assigns the specified facet to this canvas.

```
#define INCL_ODCANVAS
#define INCL_ODAPI
#include <os2.h>
```

```
ODFacet      *facet;
```

```
SetFacet(facet);
```

---

### SetFacet Parameter - facet

**facet** (ODFacet \*) - input

A reference to the facet to be assigned to this canvas.

---

### SetFacet - Return Value

None.

---

### SetFacet - Parameters

**facet** (ODFacet \*) - input

A reference to the facet to be assigned to this canvas.

None.

---

## SetFacet - Remarks

OpenDoc calls this method when the canvas is added to the facet.

---

## SetFacet - Related Methods

### Related Methods

- [ODCanvas::GetFacet](#)
- 

## SetFacet - Topics

### Class:

ODCanvas

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## SetOwner

---

## SetOwner - Syntax

This method assigns the specified owning part to this canvas.

```
#define INCL_ODCANVAS  
#define INCL_ODAPI  
#include <os2.h>
```

```
ODPart      *owner;
```

```
SetOwner(owner);
```

---

## SetOwner Parameter - owner

**owner** (ODPart \*) - input  
A reference to the owning part to be assigned to this canvas.

-----

## SetOwner - Return Value

None.

-----

## SetOwner - Parameters

**owner** (ODPart \*) - input  
A reference to the owning part to be assigned to this canvas.

None.

-----

## SetOwner - Remarks

You typically call this method immediately after your part creates a custom canvas.

-----

## SetOwner - Related Methods

### Related Methods

- [ODCanvas::AcquireOwner](#)

-----

## SetOwner - Topics

**Class:**  
ODCanvas

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

-----

# SetPlatformCanvas

---

## SetPlatformCanvas - Syntax

This method assigns the drawing structure for the specified graphics system to this canvas.

```
#define INCL_ODCANVAS
#define INCL_ODAPI
#include <os2.h>

ODGraphicsSystem      g;
ODPlatformCanvas      *c;

SetPlatformCanvas(g, c);
```

---

## SetPlatformCanvas Parameter - g

**g** ([ODGraphicsSystem](#)) - input

The graphics system whose drawing structure is to be set. Valid graphics systems are platform-dependent.

On OS/2, this parameter should always be set to KODGPI.

---

## SetPlatformCanvas Parameter - c

**c** ([ODPlatformCanvas \\*](#)) - input

The graphics-system specific drawing structure to be assigned to this canvas or KODNULL to remove the drawing structure from a graphics system. Valid values for this parameter are graphics-system-dependent.

On OS/2, [ODPlatformCanvas](#) is a SOM class which encapsulates one or more GPI presentation spaces.

---

## SetPlatformCanvas - Return Value

None.

---

## SetPlatformCanvas - Parameters

**g** ([ODGraphicsSystem](#)) - input

The graphics system whose drawing structure is to be set. Valid graphics systems are platform-dependent.

On OS/2, this parameter should always be set to kODGPI.

**c** ([ODPlatformCanvas \\*](#)) - input

The graphics-system specific drawing structure to be assigned to this canvas or kODNULL to remove the drawing structure from a graphics system. Valid values for this parameter are graphics-system-dependent.

On OS/2, [ODPlatformCanvas](#) is a SOM class which encapsulates one or more GPI presentation spaces.

None.

---

## SetPlatformCanvas - Remarks

You can assign any graphics-system-specific drawing structure that a part might use. On some platforms, a canvas can have drawing structures for two or more graphics systems simultaneously.

---

## SetPlatformCanvas - Exception Handling

kODErrInvalidGraphicsSystem

The implementation of OpenDoc does not support the specified graphics system or that graphics system is not installed or available.

---

## SetPlatformCanvas - Related Methods

### Related Methods

- [ODCanvas::GetPlatformCanvas](#)
- [ODCanvas::HasPlatformCanvas](#)

---

## SetPlatformCanvas - Topics

### Class:

[ODCanvas](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

# SetPlatformPrintJob

---

## SetPlatformPrintJob - Syntax

This method assigns the print job for the specified graphics system to this canvas.

```
#define INCL_ODCANVAS
#define INCL_ODAPI
#include <os2.h>

ODGraphicsSystem    g;
ODPlatformPrintJob  j;

SetPlatformPrintJob(g, j);
```

---

## SetPlatformPrintJob Parameter - g

**g** ([ODGraphicsSystem](#)) - input

The graphics system whose print job is to be set. Valid graphics systems are platform-dependent.

On OS/2, this parameter should always be set to KODGPI.

---

## SetPlatformPrintJob Parameter - j

**j** ([ODPlatformPrintJob](#)) - input

The graphics-system-specific print job to be assigned to this canvas or KODNULL to clear the print job. Valid values for this parameter are graphics-system-dependent.

On OS/2, this type is a pointer to a [PRINTDEST](#) structure. The caller should not modify or free the memory in this structure until after the print job has been spooled and the platform print job has been cleared by calling the SetPlatformPrintJob method with this parameter set to KODNULL.

---

## SetPlatformPrintJob - Return Value

None.

---

## SetPlatformPrintJob - Parameters



**g** ([ODGraphicsSystem](#)) - input

The graphics system whose print job is to be set. Valid graphics systems are platform-dependent.

On OS/2, this parameter should always be set to KODGPI.

**j** ([ODPlatformPrintJob](#)) - input

The graphics-system-specific print job to be assigned to this canvas or KODNULL to clear the print job. Valid values for this parameter are graphics-system-dependent.

On OS/2, this type is a pointer to a [PRINTDEST](#) structure. The caller should not modify or free the memory in this structure until after the print job has been spooled and the platform print job has been cleared by calling the `SetPlatformPrintJob` method with this parameter set to KODNULL.

None.

-----

## SetPlatformPrintJob - Remarks

You need to call this method only when you are creating a static canvas to be used as a print job.

-----

## SetPlatformPrintJob - Related Methods

### Related Methods

- [ODCanvas::GetPlatformPrintJob](#)
- [ODCanvas::HasPlatformPrintJob](#)

-----

## SetPlatformPrintJob - Topics

### Class:

[ODCanvas](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

-----

## Validate

-----

## Validate - Syntax

This method subtracts the specified area from the update shape for this canvas.

```
#define INCL_ODCANVAS
#define INCL_ODAPI
#include <os2.h>

ODShape      *shape;

Validate(shape);
```

---

## Validate Parameter - shape

**shape** (ODShape \*) - input

A reference to the shape object defining the area to be subtracted from the update shape for this canvas.

---

## Validate - Return Value

None.

---

## Validate - Parameters

**shape** (ODShape \*) - input

A reference to the shape object defining the area to be subtracted from the update shape for this canvas.

None.

---

## Validate - Remarks

OpenDoc calls this method internally to mark an area of a canvas that no longer needs updating. Your part typically calls its facet's [Validate](#) method instead of this method because that method transforms and clips the shape from the coordinate space of the facet to the coordinate space of its canvas.

---

## Validate - Related Methods

### Related Methods

- [ODCanvas::Invalidate](#)
- [ODFacet::Validate](#)

---

# Validate - Topics

## Class:

ODCanvas

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

# ODClipboard

**Class Definition File:** CLIPBD.IDL

## Class Hierarchy

SOMObject

  ODObject

    ODBaseClipboard

**ODClipboard**

## Description

An object of the ODClipboard class provides data-transfer services between OpenDoc documents and their parts and between OpenDoc and non-OpenDoc documents.

On each platform, OpenDoc provides an implementation of the ODClipboard class that gives a common, platform-independent clipboard interface for all parts. Each such implementation of ODClipboard provides access to a platform-specific clipboard (also known as the *system clipboard*) through OpenDoc storage units. This approach not only shields part editors from the underlying system clipboard mechanism, it also allows for more complex data transfers between different parts and documents, including linking.

When a document is opened, the session object creates a single clipboard object. All parts of that document share the clipboard object; you can obtain a reference to it by calling the session object's [GetClipboard](#) method. You must not cache this object; instead, you must call the session object's [GetClipboard](#) method whenever you need the clipboard object.

Parts typically invoke ODClipboard methods in response to **Edit** menu commands, such as **Copy**, **Cut**, **Paste**, and **Paste As**, and during certain linking and drag-and-drop operations. In addition to the traditional data-transfer features associated with these commands, OpenDoc's ability to clone objects allows a data-transfer operation to involve not only intrinsic content of the source part, but also the content of embedded parts, which may be of any part kind and embedded to any depth. The OpenDoc clipboard mechanism also allows more flexibility in how items are pasted. For example, the transferred data can be embedded or incorporated into the destination part with optional translations.

You should access the clipboard only when your part is running in the front-most process and your part owns clipboard focus. You can acquire the clipboard focus by calling the arbitrator object's [RequestFocus](#) method. Acquiring the clipboard focus ensures that no other part can access or modify the data while your transfers are taking place. An active part must, therefore, acquire the clipboard focus before enabling the appropriate commands in the **Edit** menu. The recommended strategy is to acquire focus in your part's [AdjustMenus](#) method and to relinquish focus in your part's [HandleEvent](#) method.

Once your part has the clipboard focus, you can write data to, or read data from, the clipboard. If you are writing data to the clipboard, you must first call the [Clear](#) method of the clipboard object. Data transfers requested by any part operate on the clipboard object's content storage unit. You obtain a reference to that storage unit by calling the clipboard's [GetContentStorageUnit](#) method. The content storage unit contains the data most recently cut or copied to the clipboard. You must not cache the storage unit; instead, you must call the [GetContentStorageUnit](#) method whenever you need to access the storage unit.

To transfer persistent objects to or from the clipboard, you must clone them using the cloning methods of their draft object. Additional methods of ODClipboard transfer data between the system clipboard and the clipboard object's storage unit.

For efficient transfer of large amounts of data to and from the clipboard, OpenDoc supports *promises*. The source part can delay a data transfer by writing a promise to the clipboard. If, and only if, a destination subsequently seeks to retrieve the data, the source part fulfills its promise by writing the data to the clipboard. This time-saving technique is transparent to the destination part.

The [GetUpdateID](#) method of the clipboard object returns an update ID that uniquely identifies the current generation or version of the clipboard. You can save this update ID and use it to detect subsequent changes in the clipboard content.

Parts that support cut and paste must support undo of those operations. If the user moves objects from one part to another and then undoes the move, the objects must be reinstated at the source. This can happen only if the part initiating the cut and the part performing the paste both support undo. Your part should notify the clipboard object whenever a cut, copy, or paste operation is done, redone, or undone by calling the clipboard object's [ActionDone](#), [ActionRedone](#), or [ActionUndone](#) method, respectively.

For more information about cloning, promises, and using the clipboard object, see the chapter on data transfer in the *OpenDoc Programming Guide*. For more information on cloning methods, see the description of the classes [ODDraft](#) and [ODPart](#). For information on undoing and redoing actions, see the chapter on windows and menus in the *OpenDoc Programming Guide* and the class description for [ODUndo](#). For information on using a link specification to indicate that the source part can create a link, see the class description for [ODLinkSpec](#).

## OS/2 Implementation Considerations

### Standard OS/2 Clipboard Formats Support

Data transfer for standard OS/2 clipboard formats is achieved using the standard OS/2 kinds as value types of the `kODPropContents` property in the clipboard object's storage unit.

The following is a list of the supported standard OS/2 formats and their associated kinds:

Format	Standard Kind
CF_BITMAP	kODKindOS2Bitmap
CF_DSPBITMAP	kODKindOS2DspBitmap
CF_DSPMETAFILE	kODKindOS2DspMetafile
CF_DSPTEXT	kODKindOS2DspText
CF_METAFILE	kODKindOS2Metafile
CF_PALETTE	kODKindOS2Palette
CF_TEXT	kODKindOS2Text

### Private OS/2 Clipboard Formats

In order to insure proper data transfer, private clipboard formats must be registered with the `ODClipboard` object prior to any reading or writing operations, by calling the [RegisterClipboardFormat](#) method. A deregistration method, [DeregisterClipboardFormat](#), is also provided.

### OS/2 Clipboard Formats and Part Registration

Parts that support any of the above standard OS/2 clipboard formats should register the corresponding standard kind. Parts that support private OS/2 clipboard formats should register the format names as kinds. This ensures that if non-OpenDoc content of a certain type is pasted into a container part, the appropriate part editor is chosen to handle the operation.

## Methods

The methods defined by the `ODClipboard` class include:

- [ActionDone](#)
- [ActionRedone](#)
- [ActionUndone](#)
- [CanEmbed](#)
- [CanIncorporate](#)
- [Clear](#)
- [DeregisterClipboardFormat](#)
- [DraftClosing](#)
- [DraftSaved](#)
- [ExportClipboard](#)
- [GetContentStorageUnit](#)
- [GetUpdateID](#)
- [RegisterClipboardFormat](#)
- [SetPlatformClipboard](#)
- [ShowPasteAsDialog](#)

## Overridden Methods

There are currently no methods overridden by the `ODClipboard` class.

-----

# ActionDone

---

# ActionDone - Syntax

This method notifies this clipboard that a cut, copy, or paste action was done.

```
#define INCL_ODCLIPBOARD
#define INCL_ODAPI
#include <os2.h>

ODCloneKind    cloneKind;
ODUpdateID     rv;

rv = ActionDone(cloneKind);
```

---

## ActionDone Parameter - cloneKind

**cloneKind** ([ODCloneKind](#)) - input

The type of clone operation that copied data to or from the clipboard. This parameter can be set to one of the following values:

- [kODCloneCopy](#) Copy into the content storage unit of the clipboard object or the drag-and-drop object.
- [kODCloneCut](#) Cut into the content storage unit of the clipboard object or the drag-and-drop object.
- [kODClonePaste](#) Paste from the content storage unit of the clipboard object or the drag-and-drop object.

---

## ActionDone Return Value - rv

**rv** ([ODUpdateID](#)) - returns

The update ID identifying the version of the clipboard content involved with this operation.

---

## ActionDone - Parameters

**cloneKind** ([ODCloneKind](#)) - input

The type of clone operation that copied data to or from the clipboard. This parameter can be set to one of the following values:

- [kODCloneCopy](#) Copy into the content storage unit of the clipboard object or the drag-and-drop object.
- [kODCloneCut](#) Cut into the content storage unit of the clipboard object or the drag-and-drop object.
- [kODClonePaste](#) Paste from the content storage unit of the clipboard object or the drag-and-drop object.

**rv** ([ODUpdateID](#)) - returns

The update ID identifying the version of the clipboard content involved with this operation.

---

## ActionDone - Remarks

Your part should call this method whenever it performs a cut, copy, or paste operation. You should cache the returned update ID in case you ever need to undo or redo the action.

---

## ActionDone - Exception Handling

kODErrIllegalClipboardCloneKind

The specified clone kind was not kODCloneCopy, kODCloneCut, or kODClonePaste.

---

## ActionDone - Related Methods

### Related Methods

- [ODClipboard::ActionRedone](#)
  - [ODClipboard::ActionUndone](#)
  - [ODPart::RedoAction](#)
  - [ODPart::UndoAction](#)
- 

## ActionDone - Topics

### Class:

[ODClipboard](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## ActionRedone

---

## ActionRedone - Syntax

This method notifies this clipboard that a cut, copy, or paste action was redone.

```
#define INCL_ODCLIPBOARD
#define INCL_ODAPI
#include <os2.h>

ODUpdateID    update;
ODCloneKind   originalCloneKind;

ActionRedone(update, originalCloneKind);
```

---

## ActionRedone Parameter - update

**update** ([ODUpdateID](#)) - input  
The update ID identifying the version of the clipboard content involved with this operation.

---

## ActionRedone Parameter - originalCloneKind

**originalCloneKind** ([ODCloneKind](#)) - input  
The type of clone operation that copied data to or from the clipboard. This parameter can be set to one of the following values:

- kODCloneCopy      Copy into the content storage unit of the clipboard object or the drag-and-drop object.
- kODCloneCut        Cut into the content storage unit of the clipboard object or the drag-and-drop object.
- kODClonePaste      Paste from the content storage unit of the clipboard object or the drag-and-drop object.

---

## ActionRedone - Return Value

None.

---

## ActionRedone - Parameters

**update** ([ODUpdateID](#)) - input  
The update ID identifying the version of the clipboard content involved with this operation.

**originalCloneKind** ([ODCloneKind](#)) - input  
The type of clone operation that copied data to or from the clipboard. This parameter can be set to one of the following values:

- kODCloneCopy      Copy into the content storage unit of the clipboard object or the drag-and-drop object.
- kODCloneCut

Cut into the content storage unit of the clipboard object or the drag-and-drop object.

kODClonePaste

Paste from the content storage unit of the clipboard object or the drag-and-drop object.

None.

-----

## ActionRedone - Remarks

Your part should call this method whenever it redoes a cut, copy, or paste operation. The update ID should be the ID returned by the [ActionDone](#) method at the time your part originally performed the action. The clone kind should be the same clone kind as specified in that call to the [ActionDone](#) method.

-----

## ActionRedone - Exception Handling

kODErrIllegalClipboardCloneKind

The specified clone kind was not kODCloneCopy, kODCloneCut, or kODClonePaste.

-----

## ActionRedone - Related Methods

### Related Methods

- [ODClipboard::ActionDone](#)
- [ODClipboard::ActionUndone](#)
- [ODPart::RedoAction](#)

-----

## ActionRedone - Topics

### Class:

[ODClipboard](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

-----

## ActionUndone



---

# ActionUndone - Syntax

This method notifies this clipboard that a cut, copy, or paste action was undone.

```
#define INCL_ODCLIPBOARD
#define INCL_ODAPI
#include <os2.h>

ODUpdateID    update;
ODCloneKind   originalCloneKind;

ActionUndone(update, originalCloneKind);
```

---

## ActionUndone Parameter - update

**update** ([ODUpdateID](#)) - input

The update ID identifying the version of the clipboard content involved with this operation.

---

## ActionUndone Parameter - originalCloneKind

**originalCloneKind** ([ODCloneKind](#)) - input

The type of clone operation that copied data to or from the clipboard. This parameter can be set to one of the following values:

kODCloneCopy

Copy into the content storage unit of the clipboard object or the drag-and-drop object.

kODCloneCut

Cut into the content storage unit of the clipboard object or the drag-and-drop object.

kODClonePaste

Paste from the content storage unit of the clipboard object or the drag-and-drop object.

---

## ActionUndone - Return Value

None.

---

## ActionUndone - Parameters

**update** ([ODUpdateID](#)) - input

The update ID identifying the version of the clipboard content involved with this operation.

**originalCloneKind** ([ODCloneKind](#)) - input

The type of clone operation that copied data to or from the clipboard. This parameter can be set to one of the following values:

kODCloneCopy

Copy into the content storage unit of the clipboard object or the drag-and-drop object.

kODCloneCut

Cut into the content storage unit of the clipboard object or the drag-and-drop object.

kODClonePaste

Paste from the content storage unit of the clipboard object or the drag-and-drop object.

None.

---

## ActionUndone - Remarks

Your part should call this method whenever it undoes a cut, copy, or paste operation. The update ID should be the ID returned by the [ActionDone](#) method at the time your part originally performed the action. The clone kind should be the same clone kind as specified in that call to the [ActionDone](#) method.

---

## ActionUndone - Exception Handling

kODErrIllegalClipboardCloneKind

The specified clone kind was not kODCloneCopy, kODCloneCut, or kODClonePaste.

---

## ActionUndone - Related Methods

### Related Methods

- [ODClipboard::ActionDone](#)
  - [ODClipboard::ActionRedone](#)
  - [ODPart::UndoAction](#)
- 

## ActionUndone - Topics

### Class:

ODClipboard

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

# CanEmbed (OS/2)

---

## CanEmbed (OS/2) - Syntax

This method queries the registration manager for a part editor capable of handling the content of the clipboard storage unit.

```
#define INCL_ODCLIPBOARD
#define INCL_ODAPI
#include <os2.h>

ODBoolean    rv;

rv = CanEmbed();
```

---

## CanEmbed (OS/2) Return Value - rv

**rv** (ODBoolean) - returns

A flag indicating whether a part editor was found.

kODTrue

A part editor was found.

kODFalse

None of the registered part editors can handle the clipboard content.

---

## CanEmbed (OS/2) - Parameters

**rv** (ODBoolean) - returns

A flag indicating whether a part editor was found.

kODTrue

A part editor was found.

kODFalse

None of the registered part editors can handle the clipboard content.

---

## CanEmbed (OS/2) - Remarks

This method is called by container parts to determine whether to enable the **Paste** menu item.

---

## CanEmbed (OS/2) - Topics

**Class:**

ODClipboard

Select an item:

[Syntax](#)[Parameters](#)[Returns](#)[Remarks](#)

---

## CanIncorporate (OS/2)

---

### CanIncorporate (OS/2) - Syntax

This method determines if there is a format on the PM clipboard matching the supplied kind.

```
#define INCL_ODCLIPBOARD
#define INCL_ODAPI
#include <os2.h>

ODType      kind;
ODBoolean    rv;

rv = CanIncorporate(kind);
```

---

### CanIncorporate (OS/2) Parameter - kind

**kind** ([ODType](#)) - input

An OpenDoc part kind, a standard OS/2 kind, or a private OS/2 clipboard format name.

---

### CanIncorporate (OS/2) Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating whether a match was found.

kODTrue

A match was found.

kODFalse

A match was not found.

---

### CanIncorporate (OS/2) - Parameters

**kind** ([ODType](#)) - input

An OpenDoc part kind, a standard OS/2 kind, or a private OS/2 clipboard format name.

**rv** ([ODBoolean](#)) - returns

A flag indicating whether a match was found.

kODTrue

A match was found.

kODFalse

A match was not found.

-----

## CanIncorporate (OS/2) - Remarks

This method is called by non-container parts for the kinds that it supports to determine whether to enable the **Paste** menu item.

-----

## CanIncorporate (OS/2) - Topics

### Class:

ODClipboard

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

-----

## Clear

-----

## Clear - Syntax

This method immediately removes all data stored in this clipboard object.

```
#define INCL_ODCLIPBOARD
#define INCL_ODAPI
#include <os2.h>
```

```
Clear();
```

-----

## Clear - Return Value

None.

---

## Clear - Parameters

None.

---

## Clear - Remarks

For thread-safe operations, the active part must acquire the clipboard focus before invoking this method. This method also causes the system clipboard to be cleared, but at a time that dependant on both the platform and the document shell.

The object calling this method must not be holding a storage-unit reference returned by a prior call to the [GetContentStorageUnit](#) method.

After this method executes successfully, the next call to the [GetContentStorageUnit](#) method returns a storage-unit object with no properties.

---

## Clear - Related Methods

### Related Methods

- [ODArbitrator::RequestFocus](#)
  - [ODClipboard::GetContentStorageUnit](#)
- 

## Clear - Topics

### Class:

ODClipboard

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## DeregisterClipboardFormat (OS/2)

---

## DeregisterClipboardFormat (OS/2) - Syntax

This method deregisters an OS/2 private clipboard format which was previously registered with this clipboard object through a call to the [RegisterClipboardFormat](#) method.

```
#define INCL_ODCLIPBOARD
#define INCL_ODAPI
#include <os2.h>

ODType      odType;
ODBoolean    rv;

rv = DeregisterClipboardFormat(odType);
```

-----

## DeregisterClipboardFormat (OS/2) Parameter - odType

**odType** ([ODType](#)) - input  
The name of the OS/2 private clipboard format to be deregistered.

-----

## DeregisterClipboardFormat (OS/2) Return Value - rv

**rv** ([ODBoolean](#)) - returns  
A flag indicating whether the format was successfully deregistered.

kODTrue	The format was successfully deregistered.
kODFalse	The specified format could not be found among the formats registered with this clipboard object or an error occurred.

-----

## DeregisterClipboardFormat (OS/2) - Parameters

**odType** ([ODType](#)) - input  
The name of the OS/2 private clipboard format to be deregistered.

**rv** ([ODBoolean](#)) - returns  
A flag indicating whether the format was successfully deregistered.

kODTrue	The format was successfully deregistered.
kODFalse	The specified format could not be found among the formats registered with this clipboard object or an error occurred.

-----

## DeregisterClipboardFormat (OS/2) - Remarks

It is not mandatory to call this method for every private format you register because all registered private formats are automatically deregistered when the clipboard object is destroyed.

---

# DeregisterClipboardFormat (OS/2) - Related Methods

## Related Methods

- [ODClipboard::RegisterClipboardFormat](#)

---

# DeregisterClipboardFormat (OS/2) - Topics

## Class:

ODClipboard

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

# DraftClosing

---

## DraftClosing - Syntax

This method is called by the document shell or a container application to notify this clipboard object that the specified draft is about to be closed or reverted.

```
#define INCL_ODCLIPBOARD
#define INCL_ODAPI
#include <os2.h>

ODDraft      *draft;

DraftClosing(draft);
```

---

## DraftClosing Parameter - draft

**draft** (ODDraft \*) - input

A reference to the draft to be closed or reverted.

---

## DraftClosing - Return Value



None.

---

## DraftClosing - Parameters

**draft** (ODDraft \*) - input

A reference to the draft to be closed or reverted.

None.

---

## DraftClosing - Remarks

The document shell or a container application calls this method when the specified draft is about to be closed or reverted. Parts must not call this method.

This method can be used to ensure the integrity of this clipboard object's reference to the destination draft object involved in paste operations. If the specified draft is closed or reverted, the clipboard's reference to the draft becomes invalid and must be deleted to avoid the possible corruption of destination draft objects.

---

## DraftClosing - Topics

### Class:

ODClipboard

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

## DraftSaved

---

## DraftSaved - Syntax

This method is called by the document shell or a container application to notify this clipboard object that the specified draft has been saved.

```
#define INCL_ODCLIPBOARD
#define INCL_ODAPI
#include <os2.h>
```

```
ODDraft      *draft;  
DraftSaved(draft);
```

---

## DraftSaved Parameter - draft

**draft** (ODDraft \*) - input  
A reference to the draft that was saved.

---

## DraftSaved - Return Value

None.

---

## DraftSaved - Parameters

**draft** (ODDraft \*) - input  
A reference to the draft that was saved.

None.

---

## DraftSaved - Remarks

The document shell calls this method after saving every draft. Parts must not call this method.

Any objects cut from the saved draft that are still on the clipboard must be treated as if they were copied if they are later pasted back into the saved draft.

---

## DraftSaved - Topics

### Class:

ODClipboard

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

# ExportClipboard

---

## ExportClipboard - Syntax

This method is called by the document shell or container application to update the system clipboard if this clipboard object has been changed since the last update of the system clipboard.

```
#define INCL_ODCLIPBOARD
#define INCL_ODAPI
#include <os2.h>
```

```
ExportClipboard();
```

---

## ExportClipboard - Return Value

None.

---

## ExportClipboard - Parameters

None.

---

## ExportClipboard - Remarks

OpenDoc calls this method internally. Parts must not call this method, but can instead call the [SetPlatformClipboard](#) method.

After this method executes successfully, the system clipboard contains the most recently cut or copied items from this clipboard object.

---

## ExportClipboard - Exception Handling

`kODErrOutOfMemory`

There is not enough memory to complete the operation.

This method may also throw platform-specific exceptions.

---

# ExportClipboard - Related Methods

## Related Methods

- [ODClipboard::SetPlatformClipboard](#)
- 

# ExportClipboard - Topics

## Class:

[ODClipboard](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

# GetContentStorageUnit

---

## GetContentStorageUnit - Syntax

This method returns a reference to the storage unit containing this clipboard object's current content.

```
#define INCL_ODCLIPBOARD
#define INCL_ODAPI
#include <os2.h>

ODStorageUnit      *rv;

rv = GetContentStorageUnit();
```

---

## GetContentStorageUnit Return Value - rv

**rv** (ODStorageUnit \*) - returns

A reference to the storage-unit object containing the content of the clipboard.

---

## GetContentStorageUnit - Parameters

**rv** (ODStorageUnit \*) - returns  
A reference to the storage-unit object containing the content of the clipboard.

---

## GetContentStorageUnit - Remarks

You can read data from or write data to the returned storage unit. You must not cache the returned storage unit; instead, you must call this method whenever you need to access the content storage unit.

You must acquire the clipboard focus before calling this method and relinquish the focus after the data transfer. If you are writing data, you must also call the [Clear](#) method before calling this method.

The clipboard object handles the creation and destruction of its content storage unit, so you must neither dispose of nor release the returned storage unit.

---

## GetContentStorageUnit - Exception Handling

kODErrOutOfMemory	There is not enough memory to complete the operation.
-------------------	---

---

## GetContentStorageUnit - Related Methods

### Related Methods

- [ODClipboard::Clear](#)
  - [ODClipboard::GetUpdateID](#)
- 

## GetContentStorageUnit - Topics

### Class:

ODClipboard

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## GetUpdateID

---

# GetUpdateID - Syntax

This method returns the update ID identifying the current generation or version of this clipboard object.

```
#define INCL_ODCLIPBOARD
#define INCL_ODAPI
#include <os2.h>

ODUpdateID    rv;

rv = GetUpdateID();
```

---

## GetUpdateID Return Value - rv

**rv** ([ODUpdateID](#)) - returns  
The update ID of the current clipboard content.

---

## GetUpdateID - Parameters

**rv** ([ODUpdateID](#)) - returns  
The update ID of the current clipboard content.

---

## GetUpdateID - Remarks

Your part may call this method to check whether another part has changed the clipboard. The returned update ID uniquely identifies the current generation or version of the clipboard. Note that update ID values have no significance other than in the context of testing them for equality. Also, the update ID value returned by this method is valid only during the current session.

It is possible, but dangerous, to call this method before your part has acquired the clipboard focus.

**Important:** Clipboard update IDs are used for a different purpose than link and link-source update IDs. You should never set the update ID of a link-source object to an update ID returned by this method.

---

## GetUpdateID - Related Methods

### Related Methods

- [ODClipboard::ActionDone](#)
  - [ODLink::GetUpdateID](#)
  - [ODLinkSource::GetUpdateID](#)
-

# GetUpdateID - Topics

## Class:

ODClipboard

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

## RegisterClipboardFormat (OS/2)

---

### RegisterClipboardFormat (OS/2) - Syntax

This method registers an OS/2 private clipboard format with this clipboard object.

```
#define INCL_ODCLIPBOARD
#define INCL_ODAPI
#include <os2.h>

ODType          odType;
ODPlatformType  odPlatformType;
ODULong         formatInfo;
ODBoolean       rv;

rv = RegisterClipboardFormat(odType, odPlatformType,
                             formatInfo);
```

---

### RegisterClipboardFormat (OS/2) Parameter - odType

**odType** ([ODType](#)) - input

The name of the OS/2 private clipboard format to be registered.

---

### RegisterClipboardFormat (OS/2) Parameter - odPlatformType

**odPlatformType** ([ODPlatformType](#)) - input

The format identification number.

---

# RegisterClipboardFormat (OS/2) Parameter - formatInfo

**formatInfo** (ODULong) - input

A flag indicating the type of data for the format. This parameter can be set to one of the following values:

CFI\_HANDLE

CFI\_POINTER

-----

# RegisterClipboardFormat (OS/2) Return Value - rv

**rv** (ODBoolean) - returns

A flag indicating whether the format has been successfully registered.

kODTrue

The format has been successfully registered.

kODFalse

Another format was previously registered with either the same format name or ID or an error occurred.

-----

# RegisterClipboardFormat (OS/2) - Parameters

**odType** (ODType) - input

The name of the OS/2 private clipboard format to be registered.

**odPlatformType** (ODPlatformType) - input

The format identification number.

**formatInfo** (ODULong) - input

A flag indicating the type of data for the format. This parameter can be set to one of the following values:

CFI\_HANDLE

CFI\_POINTER

**rv** (ODBoolean) - returns

A flag indicating whether the format has been successfully registered.

kODTrue

The format has been successfully registered.

kODFalse

Another format was previously registered with either the same format name or ID or an error occurred.

-----

# RegisterClipboardFormat (OS/2) - Remarks

A part should call this method for all private clipboard formats it intends to use; the formats should be registered prior to any data transfer operation from or to the system clipboard (by calling the clipboard object's [GetContentStorageUnit](#) or [ExportClipboard](#) method) that might involve the private formats. Once a format is registered with the clipboard object, it remains registered until the [DeregisterClipboardFormat](#) method is called for that format or the current OpenDoc session ends.



---

# RegisterClipboardFormat (OS/2) - Related Methods

## Related Methods

- [ODClipboard::DeregisterClipboardFormat](#)

---

# RegisterClipboardFormat (OS/2) - Topics

## Class:

[ODClipboard](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

# SetPlatformClipboard

---

## SetPlatformClipboard - Syntax

This method copies any data of the specified types from this clipboard object to the system clipboard if this clipboard object has changed since the last update to the system clipboard.

```
#define INCL_ODCLIPBOARD
#define INCL_ODAPI
#include <os2.h>

ODPlatformTypeList      *typeList;

SetPlatformClipboard(typeList);
```

---

## SetPlatformClipboard Parameter - typeList

**typeList** (ODPlatformTypeList \*) - input

A reference to the list of platform-dependent types to be copied to the system clipboard.

---

## SetPlatformClipboard - Return Value

None.

---

## SetPlatformClipboard - Parameters

**typeList** (ODPlatformTypeList \*) - input

A reference to the list of platform-dependent types to be copied to the system clipboard.

None.

---

## SetPlatformClipboard - Remarks

You must call this method prior to invoking any platform-specific service that uses the system clipboard.

This method examines the value type of each value in the content property of this clipboard's storage unit. If the value type is in the specified platform type-list or if the *typeList* parameter is KODNULL, this method copies the data in that value to the system clipboard.

To keep system-clipboard behavior consistent, this method does not attempt translation to a specified type that is not present in this clipboard object.

---

## SetPlatformClipboard - Exception Handling

KODErrOutOfMemory

There is not enough memory to complete the operation.

This method may also throw platform-specific exceptions.

---

## SetPlatformClipboard - Related Methods

### Related Methods

- [ODClipboard::ExportClipboard](#)
  - [ODPart::AdjustMenus](#)
- 

## SetPlatformClipboard - Topics

### Class:

ODClipboard

Select an item:

[Syntax](#)

---

## ShowPasteAsDialog (OS/2)

---

### ShowPasteAsDialog (OS/2) - Syntax

This method displays the Paste As dialog box and sets the appropriate dialog items according to the input parameters.

```
#define INCL_ODCLIPBOARD
#define INCL_ODAPI
#include <os2.h>

ODBoolean          canPasteLink;
ODPasteAsMergeSetting mergeSetting;
ODFacet            *facet;
ODTypeToken        viewType;
ODPasteAsResult     *theResult;
ODBoolean          rv;

rv = ShowPasteAsDialog(canPasteLink, mergeSetting,
    facet, viewType, theResult);
```

---

### ShowPasteAsDialog (OS/2) Parameter - canPasteLink

**canPasteLink** ([ODBoolean](#)) - input

A flag indicating whether the destination part allows a link to be created. This parameter can be set to one of the following values:

kODTrue	The destination part allows a link to be created.
kODFalse	A link cannot be created.

---

### ShowPasteAsDialog (OS/2) Parameter - mergeSetting

**mergeSetting** ([ODPasteAsMergeSetting](#)) - input

A flag indicating whether embedding and merging are supported. The value of this parameter determines which radio button in the **At the Destination** group box is initially selected and whether the other button is available.

kODPasteAsEmbed	<b>Embed As</b> is initially selected; <b>Merge with Contents</b> is available.
kODPasteAsEmbedOnly	<b>Embed As</b> is selected; <b>Merge with Contents</b> is disabled.
kODPasteAsMerge	

**Merge with Contents** is initially selected; **Embed As** is available.  
kODPasteAsMergeOnly  
**Merge with Contents** is selected; **Embed As** is disabled.

---

## ShowPasteAsDialog (OS/2) Parameter - facet

**facet** (ODFacet \*) - input  
A reference to the facet from which the Paste As dialog box is triggered.

---

## ShowPasteAsDialog (OS/2) Parameter - viewType

**viewType** (ODTypeToken) - input  
A tokenized string representing the initial setting for the view type of the embedded part (if embedding is chosen).

This parameter must be the tokenized form of one of the following view-type constants. You can call the session's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODViewAsFrame	Framed view type.
kODViewAsLargeIcon	Large-icon (standard) view type
kODViewAsSmallIcon	Small-icon view type.
kODViewAsThumbNail	Thumbnail view type

---

## ShowPasteAsDialog (OS/2) Parameter - theResult

**theResult** (ODPasteAsResult \*) - output  
A structure reflecting the user's selection in the Paste As dialog box.

---

## ShowPasteAsDialog (OS/2) Return Value - rv

**rv** (ODBoolean) - returns  
A flag indicating whether the user clicked the **OK** button to leave the Paste As dialog box.

kODTrue	The user clicked the <b>OK</b> button.
kODFalse	The user canceled out of the dialog box.

---

## ShowPasteAsDialog (OS/2) - Parameters

**canPasteLink** ([ODBoolean](#)) - input

A flag indicating whether the destination part allows a link to be created. This parameter can be set to one of the following values:

kODTrue	The destination part allows a link to be created.
kODFalse	A link cannot be created.

**mergeSetting** ([ODPasteAsMergeSetting](#)) - input

A flag indicating whether embedding and merging are supported. The value of this parameter determines which radio button in the **At the Destination** group box is initially selected and whether the other button is available.

kODPasteAsEmbed	<b>Embed As</b> is initially selected; <b>Merge with Contents</b> is available.
kODPasteAsEmbedOnly	<b>Embed As</b> is selected; <b>Merge with Contents</b> is disabled.
kODPasteAsMerge	<b>Merge with Contents</b> is initially selected; <b>Embed As</b> is available.
kODPasteAsMergeOnly	<b>Merge with Contents</b> is selected; <b>Embed As</b> is disabled.

**facet** ([ODFacet \\*](#)) - input

A reference to the facet from which the Paste As dialog box is triggered.

**viewType** ([ODTypeToken](#)) - input

A tokenized string representing the initial setting for the view type of the embedded part (if embedding is chosen).

This parameter must be the tokenized form of one of the following view-type constants. You can call the session's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODViewAsFrame	Framed view type.
kODViewAsLargeIcon	Large-icon (standard) view type
kODViewAsSmallIcon	Small-icon view type.
kODViewAsThumbNail	Thumbnail view type

**theResult** ([ODPasteAsResult \\*](#)) - output

A structure reflecting the user's selection in the Paste As dialog box.

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the user clicked the **OK** button to leave the Paste As dialog box.

kODTrue	The user clicked the <b>OK</b> button.
kODFalse	The user canceled out of the dialog box.

---

## ShowPasteAsDialog (OS/2) - Remarks

You should call this method in your part's [HandleEvent](#) method to display the Paste As dialog box when the user selects **Paste As** from the **Edit** menu.

If the *canPasteLink* parameter is kODTrue, and if the content storage unit contains a link specification and the draft permissions allow writing, then the **Paste with Link** check box is checked.

If the user clicks the **OK** button in the Paste As dialog box, this method returns kODTrue and sets the fields of the *theResult* output parameter to indicate the selections the user made. You must dispose of the non-null *selectedKind*, *translateKind*, and *editor* fields of the [ODPasteAsResult](#) structure when you are finished using them.

If the user cancels out of the dialog box, this method returns kODFalse, and you do not need to take any further action.

---

## ShowPasteAsDialog (OS/2) - Exception Handling

kODErrNullFacetInput  
kODErrNullPasteAsResultInput  
kODErrOutOfMemory

The *facet* parameter is null.  
The *theResult* parameter is null.  
There is not enough memory to complete the operation.

---

## ShowPasteAsDialog (OS/2) - Related Methods

### Related Methods

- [ODSession::Tokenize](#)

---

## ShowPasteAsDialog (OS/2) - Topics

### Class:

[ODClipboard](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## ODContainer

**Class Definition File:** ODCTR.IDL

### Class Hierarchy

SOMObject  
  ODObject  
    ODRefCntObject  
      **ODContainer**

### Description

An object of the ODContainer class represents a physical container storing a collection of OpenDoc documents. The ODContainer class lets you access these documents independently of the container's physical characteristics or internal structure.

The ODContainer class is implemented differently for different platforms and storage mechanism.

The ODContainer class manipulates physical containers. Similarly, the [ODDocument](#) class manipulates documents; the [ODDraft](#) class manipulates drafts; and the [ODStorageUnit](#) class manipulates storage units. This set of related classes, ODContainer, [ODDocument](#), [ODDraft](#), and [ODStorageUnit](#), is called a *container suite*. Container-suite classes are implemented as an integrated set for each platform and storage mechanism because they work intimately with each other at many levels.

Each container object can contain one or more document objects. Each document object, in turn, can contain one or more draft objects, and each draft object can contain one or more storage unit objects. Each container, document, draft and storage unit has a unique ID.

Only the OpenDoc storage system directly creates container objects. The document shell or container application creates or accesses a container object by calling the storage-system object's [CreateContainer](#) or [AcquireContainer](#) method. The storage system ensures that there is only one container object associated with each physical container.

You can access a document of a container by calling the container's [AcquireDocument](#) method. The ODContainer class is responsible for

ensuring that there is only one document object associated with each document in the container.

For more information on how ODContainer and other container-suite classes are used, see the chapters on storage and the OpenDoc run-time features in the *OpenDoc Programming Guide* .

## Methods

The methods defined by the ODContainer class include:

- [AcquireDocument](#)
- [GetID](#)
- [GetName](#)
- [GetStorageSystem](#)
- [SetName](#)

## Overridden Methods

There are currently no methods overridden by the ODContainer class.

---

# AcquireDocument

---

## AcquireDocument - Syntax

This method is called by the document shell or container application to retrieve a reference to the document object associated with the specified document ID.

```
#define INCL_ODCONTAINER
#define INCL_ODAPI
#include <os2.h>

ODDocumentID    id;
ODDocument      *rv;

rv = AcquireDocument(id);
```

---

## AcquireDocument Parameter - id

**id** ([ODDocumentID](#)) - input  
The ID of the requested document.

---

## AcquireDocument Return Value - rv

**rv** (ODDocument \*) - returns  
A reference to the specified document object.

---

# AcquireDocument - Parameters

**id** ([ODDocumentID](#)) - input  
The ID of the requested document.

**rv** ([ODDocument \\*](#)) - returns  
A reference to the specified document object.

---

## AcquireDocument - Remarks

The document shell or container application calls this method when the user opens an OpenDoc document.

This method looks in this container object for the document object associated with the specified document ID. If such a document object exists, the object is returned. If the specified document object does not exist, it is created, initialized, and returned.

This method increments the reference count of the returned document object. When the caller has finished using that document object, it should call the document's [Release](#) method.

---

## AcquireDocument - Exception Handling

kODErrInvalidDocument

The specified document does not exist.

---

## AcquireDocument - Related Methods

### Related Methods

- [ODRefCntObject::Release](#)
- 

## AcquireDocument - Topics

**Class:**  
[ODContainer](#)

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## GetID



---

## GetID - Syntax

This method returns the container ID of this container object.

```
#define INCL_ODCONTAINER
#define INCL_ODAPI
#include <os2.h>

ODContainerID    rv;

rv = GetID();
```

---

## GetID Return Value - rv

**rv** ([ODContainerID](#)) - returns  
The container ID whose buffer contains data identifying this container object.

---

## GetID - Parameters

**rv** ([ODContainerID](#)) - returns  
The container ID whose buffer contains data identifying this container object.

---

## GetID - Remarks

Although parts can call this method, they usually do not need to know the IDs of their associated containers.

The structure of the data identifying this container depends on the type of container, which was specified when the container was created. For example, the identifier for a Bento file container specifies a file-system file; the identifier for a Bento memory container is a handle for a relocatable memory block.

---

## GetID - Topics

**Class:**  
ODContainer

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

# GetName

---

## GetName - Syntax

This method returns the name of this container object.

```
#define INCL_ODCONTAINER
#define INCL_ODAPI
#include <os2.h>

ODContainerName    rv;

rv = GetName();
```

---

## GetName Return Value - rv

**rv** ([ODContainerName](#)) - returns  
The name of this container object or an empty sequence if the container does not have name.

---

## GetName - Parameters

**rv** ([ODContainerName](#)) - returns  
The name of this container object or an empty sequence if the container does not have name.

---

## GetName - Remarks

Although parts can call this method, they usually do not need to know the names of their associated container.

If this container has a name, this method returns a copy of that name; otherwise, the *text* field of the result [ODContainerName](#) structure represents an empty sequence of characters-that is, the *text* field is an [ODByteArray](#) structure whose *\_length* field contains 0 and whose *\_buffer* field contains null pointer.

---

## GetName - Related Methods

**Related Methods**

- [ODContainer::SetName](#)

---

## GetName - Topics

### Class:

ODContainer

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## GetStorageSystem

---

## GetStorageSystem - Syntax

This method returns a reference to the storage-system object that created this container object.

```
#define INCL_ODCONTAINER
#define INCL_ODAPI
#include <os2.h>

ODStorageSystem *rv;

rv = GetStorageSystem();
```

---

## GetStorageSystem Return Value - rv

**rv** (ODStorageSystem \*) - returns

A reference to the storage-system object that created this container object.

---

## GetStorageSystem - Parameters

**rv** (ODStorageSystem \*) - returns

A reference to the storage-system object that created this container object.

---

# GetStorageSystem - Topics

## Class:

ODContainer

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

---

## SetName

---

### SetName - Syntax

This method is called by the document shell or container application to set the name of this container object.

```
#define INCL_ODCONTAINER
#define INCL_ODAPI
#include <os2.h>
```

```
ODContainerName    *name;
```

```
SetName (name);
```

---

### SetName Parameter - name

**name** ([ODContainerName](#) \*) - input  
The new name of this container

---

### SetName - Return Value

None.

---

### SetName - Parameters

**name** ([ODContainerName](#) \*) - input  
The new name of this container

None.

---

## SetName - Related Methods

### Related Methods

- [ODContainer::GetName](#)

---

## SetName - Topics

### Class:

[ODContainer](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Related Methods](#)

---

## ODDesc

**Class Definition File:** [ODDESC.IDL](#)

### Class Hierarchy

SOMObject  
  ODObject  
    **ODDesc**

### Description

An object of the ODDesc class is a wrapper for a *descriptor structure* (type [AEDesc](#)), the basic structure used for building OSA event attributes and parameters.

An OSA event descriptor structure consists of a handle to data and a descriptor type that identifies the type of data to which the handle refers. An object of the ODDesc class stores the actual data. To use the OSA Event Manager functions on an [ODDesc](#) object, you must first extract the data from it and then create a descriptor structure from that data.

The ODDesc class provides methods for placing and extracting the OSA event descriptor it contains. To convert from an ODDesc descriptor object to type [AEDesc](#), you may copy the raw data and the descriptor type using the [ODDescToAEDesc](#) and [AEDescToODDesc](#) utility routines.

For more information on OSA events and the [AEDesc](#) type, see the chapter introducing OSA events in the *Open Scripting Architecture Guide and Reference for OS/2* . For general information on scripting support in OpenDoc, see the chapter on semantic events and scripting in the *OpenDoc Programming Guide* .

### Methods

The methods defined by the ODDesc class include:

- [GetDescType](#)

- [GetRawData](#)
- [InitODDesc](#)
- [SetDescType](#)
- [SetRawData](#)

#### Overridden Methods

There are currently no methods overridden by the ODDesc class.

## GetDescType

## GetDescType - Syntax

This method returns the descriptor type of this descriptor object.

```
#define INCL_ODDESC
#define INCL_ODAPI
#include <os2.h>

ODDescType    rv;

rv = GetDescType();
```

## GetDescType Return Value - rv

**rv** ([ODDescType](#)) - returns  
The descriptor type of this descriptor object.

## GetDescType - Parameters

**rv** ([ODDescType](#)) - returns  
The descriptor type of this descriptor object.

## GetDescType - Remarks

If a descriptor object represents an OSA-event-specific descriptor, the returned data may not be interpretable without using the OSA Event Manager function.

## GetDescType - Topics

**Class:**  
ODDesc

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

## GetRawData

---

### GetRawData - Syntax

This method returns the raw data contained in this descriptor object.

```
#define INCL_ODDESC
#define INCL_ODAPI
#include <os2.h>

ODByteArray    rv;

rv = GetRawData();
```

---

### GetRawData Return Value - rv

**rv** ([ODByteArray](#)) - returns  
A byte array whose buffer contains the raw data to be retrieved.

---

### GetRawData - Parameters

**rv** ([ODByteArray](#)) - returns  
A byte array whose buffer contains the raw data to be retrieved.

---

### GetRawData - Exception Handling

kODErrOutOfMemory

There is not enough memory to allocate the byte array structure's buffer.

---

# GetRawData - Topics

## Class:

ODDesc

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Exception Handling](#)

---

## InitODDesc

---

## InitODDesc - Syntax

This method initializes this descriptor object.

```
#define INCL_ODDESC
#define INCL_ODAPI
#include <os2.h>
```

```
InitODDesc();
```

---

## InitODDesc - Return Value

None.

---

## InitODDesc - Parameters

None.

---

## InitODDesc - Remarks



There is no factory method for the [ODDesc](#) class; after creating a new descriptor object, OpenDoc or your part must call this method to initialize the new descriptor object.

---

## InitODDesc - Topics

**Class:**  
ODDesc

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

## SetDescType

---

## SetDescType - Syntax

This method sets the descriptor type of this descriptor object.

```
#define INCL_ODDESC
#define INCL_ODAPI
#include <os2.h>

ODDescType    descType;

SetDescType(descType);
```

---

## SetDescType Parameter - descType

**descType** ([ODDescType](#)) - input  
The descriptor type of this descriptor object.

---

## SetDescType - Return Value

None.

---

# SetDescType - Parameters

**descType** ([ODDescType](#)) - input  
The descriptor type of this descriptor object.

None.

---

## SetDescType - Remarks

This method may change the interpretation of the data in the descriptor object, but it does not change the raw data itself.

---

## SetDescType - Topics

**Class:**  
ODDesc

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

## SetRawData

---

## SetRawData - Syntax

This method assigns the specified raw data to this descriptor object; any previous data is deleted.

```
#define INCL_ODDESC
#define INCL_ODAPI
#include <os2.h>

ODByteArray    *data;

SetRawData(data);
```

---

## SetRawData Parameter - data

**data** ([ODByteArray \\*](#)) - input  
A byte array whose buffer contains the raw data to be stored.

---

## SetRawData - Return Value

None.

---

## SetRawData - Parameters

**data** ([ODByteArray \\*](#)) - input  
A byte array whose buffer contains the raw data to be stored.

None.

---

## SetRawData - Topics

**Class:**  
ODDesc

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## ODDescList

**Class Definition File:** ODDESLST.IDL

### Class Hierarchy

SOMObject  
  ODObject  
    ODDesc  
      **ODDescList**

### Description

An object of the ODDescList class is a wrapper for a descriptor list. A *descriptor list* is a data structure of type [AEDescList](#) defined by the data type [AEDesc](#)-that is, a descriptor list is a descriptor record whose data handle refers to a list of other descriptor records (unless it is an empty list).

For more information on OSA events and the [AEDescList](#) type, see the chapter introducing OSA events in the *Open Scripting Architecture Guide and Reference for OS/2* . For general information on scripting support in OpenDoc, see the chapter on semantic events and scripting in the *OpenDoc Programming Guide* .

### Methods

The methods defined by the ODDescList class include:

- [InitODDescList](#)

#### Overridden Methods

There are currently no methods overridden by the ODDescList class.

---

## InitODDescList

---

### InitODDescList - Syntax

This method initializes this descriptor list.

```
#define INCL_ODDESCLIST
#define INCL_ODAPI
#include <os2.h>
```

```
InitODDescList();
```

---

### InitODDescList - Return Value

None.

---

### InitODDescList - Parameters

None.

---

### InitODDescList - Remarks

There is no factory method for the [ODDescList](#) class; after creating a new descriptor list, OpenDoc or your part must call this method to initialize the new descriptor list.

---

### InitODDescList - Topics

**Class:**  
ODDescList

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

# ODDispatcher

**Class Definition File:** DISPTCH.IDL

## Class Hierarchy

SOMObject  
  ODObject  
    ODBaseDispatcher  
      **ODDispatcher**

## Description

An object of the ODDispatcher class is responsible for distributing events to part editors.

When a document is opened, the session object creates a single dispatcher object. All parts of the document share the dispatcher object; you can obtain a reference to it by calling the session object's [GetDispatcher](#) method.

The dispatcher maintains a *dispatch module dictionary*, indexed by event type. The dictionary contains at least one internal dispatch module to handle standard events (such as, mouse clicks, keystrokes, and menu commands) of a particular platform. You can extend the OpenDoc dispatching system by installing additional dispatch modules to dispatch new types of events or messages to your part editor. For more information, see the class description of [ODDispatchModule](#).

Part editors do not receive events directly from the operating system; rather, OpenDoc notifies the appropriate part editor when an event occurs. To do so, the document shell's event loop calls the dispatcher's [Dispatch](#) method to dispatch events to part editors. The dispatcher handles the events it recognizes using dispatch module objects to dispatch specific events to individual parts. The dispatcher locates the dispatch module for the specified event in its dispatch module dictionary and calls the dispatch module's [Dispatch](#) method. The [Dispatch](#) method, in turn, calls the part editor's [HandleEvent](#) method to give the part the opportunity to handle the specified event. If your part contains embedded frames, OpenDoc can also send your part editor special mouse events that occur within and on the borders of your part's embedded frames' facets. The dispatcher leaves all other events, as well as events dispatched to parts but not handled by them, to the document shell to handle. Events not handled by the document shell are ignored. For more information on event handling in OpenDoc, see the chapter on user events in the *OpenDoc Programming Guide*.

OpenDoc allows you to monitor the event stream without interfering with it. By registering a dispatch module as a *monitor* for a specified event type, OpenDoc notifies the dispatch module when an event of that type occurs. You might use a monitor in a debugging environment to monitor events and display a log of the events in a window. In general, you should install monitors and not patch the dispatch module. Patching occurs when the creator of the new dispatch modules saves the existing dispatch module and then installs its own replacement dispatch module that calls the original.

## Methods

The methods defined by the ODDispatcher class include:

- [AddDispatchModule](#)
- [AddMonitor](#)
- [Dispatch](#)
- [Exit](#)
- [GetDispatchModule](#)
- [Redispatch](#)
- [RemoveDispatchModule](#)
- [RemoveMonitor](#)
- [ShouldExit](#)

## Overridden Methods

There are currently no methods overridden by the ODDispatcher class.

---

# AddDispatchModule

---

## AddDispatchModule - Syntax

This method adds the specified dispatch module for the specified event type to the dispatch module dictionary.

```
#define INCL_ODDISPATCHER
#define INCL_ODAPI
#include <os2.h>

ODEventType      eventType;
ODDispatchModule *dispatchModule;

AddDispatchModule(eventType, dispatchModule);
```

---

## AddDispatchModule Parameter - eventType

**eventType** (ODEventType) - input

A platform-specific event code that specifies the type of event to be handled by the new dispatch module. On OS/2, this is a PM message ID.

---

## AddDispatchModule Parameter - dispatchModule

**dispatchModule** (ODDispatchModule \*) - input

A reference to a dispatch module to be added.

---

## AddDispatchModule - Return Value

None.

---

## AddDispatchModule - Parameters

**eventType** (ODEventType) - input

A platform-specific event code that specifies the type of event to be handled by the new dispatch module. On OS/2, this is a PM message ID.

**dispatchModule** (ODDispatchModule \*) - input

A reference to a dispatch module to be added.

None.

---

## AddDispatchModule - Remarks

Your part editor calls this method to install a custom dispatch module. This method is not called by part editors under normal circumstances.

---

## AddDispatchModule - Exception Handling

kODErrIllegalNullDispatchModuleInput

The *dispatchModule* parameter is null.

---

## AddDispatchModule - Related Methods

### Related Methods

- [ODDispatcher::GetDispatchModule](#)
  - [ODDispatcher::RemoveDispatchModule](#)
- 

## AddDispatchModule - Topics

### Class:

ODDispatcher

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## AddMonitor

---

## AddMonitor - Syntax

This method adds the specified dispatch module, which is a monitor for the specified event type, to the dispatch module dictionary.

```

#define INCL_ODDISPATCHER
#define INCL_ODAPI
#include <os2.h>

ODEventType      eventType;
ODDispatchModule *dispatchModule;

AddMonitor(eventType, dispatchModule);

```

-----

## AddMonitor Parameter - eventType

**eventType** (ODEventType) - input

A platform-specific event code that specifies the type of event to be handled by the new dispatch module. On OS/2, this is a PM message ID.

-----

## AddMonitor Parameter - dispatchModule

**dispatchModule** (ODDispatchModule \*) - input

A reference to the dispatch module to be added.

-----

## AddMonitor - Return Value

None.

-----

## AddMonitor - Parameters

**eventType** (ODEventType) - input

A platform-specific event code that specifies the type of event to be handled by the new dispatch module. On OS/2, this is a PM message ID.

**dispatchModule** (ODDispatchModule \*) - input

A reference to the dispatch module to be added.

None.

-----

## AddMonitor - Remarks

This method is not called by part editors under normal circumstances; however, your part editor calls this method to install a custom dispatch



module as a monitor for the specified event type.

---

## AddMonitor - Exception Handling

kODErrIllegalNullDispatchModuleInput

The *dispatchModule* parameter is null.

---

## AddMonitor - Related Methods

### Related Methods

- [ODDispatcher::RemoveMonitor](#)
- 

## AddMonitor - Topics

### Class:

ODDispatcher

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

## Dispatch

---

## Dispatch - Syntax

This method is called by the document shell and container applications to dispatch the specified event to the appropriate part.

```
#define INCL_ODDISPATCHER
#define INCL_ODAPI
#include <os2.h>

ODEventData      *eventData;
ODBoolean         rv;

rv = Dispatch(eventData);
```

---

# Dispatch Parameter - eventData

**eventData** ([ODEventData \\*](#)) - in/out

A platform-specific structure representing an event. On return, the fields of this structure may have been modified.

---

## Dispatch Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the event was handled by a part.

kODTrue

The event was handled by a part.

kODFalse

The event is not associated with an existing dispatch module or the event was not handled.

---

## Dispatch - Parameters

**eventData** ([ODEventData \\*](#)) - in/out

A platform-specific structure representing an event. On return, the fields of this structure may have been modified.

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the event was handled by a part.

kODTrue

The event was handled by a part.

kODFalse

The event is not associated with an existing dispatch module or the event was not handled.

---

## Dispatch - Remarks

The document shell and container applications call this method when an event occurs. This method looks up the dispatch module for the specified event in the dispatch module dictionary and then calls the dispatch module's [Dispatch](#) method. This method can also be called by parts that handle events in dialog boxes.

---

## Dispatch - Related Methods

### Related Methods

- [ODDispatchModule::Dispatch](#)
  - [ODPart::HandleEvent](#)
-

# Dispatch - Topics

## Class:

ODDDispatcher

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

## Exit

---

### Exit - Syntax

This method is called by the document shell to set a boolean value in the dispatcher that specifies that the document shell should terminate.

```
#define INCL_ODDISPATCHER
#define INCL_ODAPI
#include <os2.h>
```

```
Exit();
```

---

### Exit - Return Value

None.

---

### Exit - Parameters

None.

---

### Exit - Remarks

This method is not called by most parts.

---

# Exit - Related Methods

## Related Methods

- [ODDDispatcher::ShouldExit](#)
- 

# Exit - Topics

## Class:

[ODDDispatcher](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

# GetDispatchModule

---

# GetDispatchModule - Syntax

This method returns a reference to the dispatch module for the specified event type.

```
#define INCL_ODDISPATCHER
#define INCL_ODAPI
#include <os2.h>

ODEventType      eventType;
ODDispatchModule *rv;

rv = GetDispatchModule(eventType);
```

---

# GetDispatchModule Parameter - eventType

**eventType** ([ODEventType](#)) - input

A platform-specific event code that specifies the type of event to be handled by the requested dispatch module. On OS/2, this is a PM message ID.

---

# GetDispatchModule Return Value - rv

**rv** (ODDispatchModule \*) - returns

A reference to the dispatch module for the specified event type or KODNULL if the event type is not associated with an existing dispatch module.

---

## GetDispatchModule - Parameters

**eventType** (ODEventType) - input

A platform-specific event code that specifies the type of event to be handled by the requested dispatch module. On OS/2, this is a PM message ID.

**rv** (ODDispatchModule \*) - returns

A reference to the dispatch module for the specified event type or KODNULL if the event type is not associated with an existing dispatch module.

---

## GetDispatchModule - Remarks

Though not highly recommended, this method might be called to patch a dispatch module. Patching occurs when the creator of the new dispatch module saves the existing dispatch module and then installs its own replacement dispatch module that calls the original. Your part editor calls this method to save the existing dispatch module so it can delegate to that dispatch module when necessary; this method is not called by parts.

---

## GetDispatchModule - Related Methods

### Related Methods

- [ODDispatcher::AddDispatchModule](#)
  - [ODDispatcher::RemoveDispatchModule](#)
- 

## GetDispatchModule - Topics

### Class:

ODDispatcher

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## Redispatch

---

# Redispatch - Syntax

This method redispatches the specified event to the appropriate part.

```
#define INCL_ODDISPATCHER
#define INCL_ODAPI
#include <os2.h>

ODEventData      *eventData;
ODEventInfo      *eventInfo;
ODBoolean        rv;

rv = Redispatch(eventData, eventInfo);
```

---

## Redispatch Parameter - eventData

**eventData** ([ODEventData \\*](#)) - in/out

A platform-specific structure representing an event. On return, the fields of this structure may have been modified.

---

## Redispatch Parameter - eventInfo

**eventInfo** ([ODEventInfo \\*](#)) - in/out

A platform-specific structure that contains additional event information. On return, the fields of this structure are filled in if the event was handled.

---

## Redispatch Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the event was handled by a part.

kODTrue

The event was handled by a part.

kODFalse

The event is not associated with an existing part or the event was not handled.

---

## Redispatch - Parameters

**eventData** ([ODEventData \\*](#)) - in/out

A platform-specific structure representing an event. On return, the fields of this structure may have been modified.

**eventInfo** ([ODEventInfo \\*](#)) - in/out

A platform-specific structure that contains additional event information. On return, the fields of this structure are filled in if the event was handled.

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the event was handled by a part.

kODTrue

The event was handled by a part.

kODFalse

The event is not associated with an existing part or the event was not handled.

-----

## Redispatch - Remarks

OpenDoc calls this method when it translates an event. For example, when the standard dispatch module transforms a mouse-down event in the menu bar to a menu event, the dispatch module redispatches the event so that monitors and patches can intercept the new event.

-----

## Redispatch - Related Methods

### Related Methods

- [ODDispatcher::Dispatch](#)
- [ODDispatchModule::Dispatch](#)

-----

## Redispatch - Topics

### Class:

[ODDispatcher](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

-----

## RemoveDispatchModule

-----

## RemoveDispatchModule - Syntax

This method removes the dispatch module for the specified event type from the dispatch module dictionary.

```
#define INCL_ODDISPATCHER
#define INCL_ODAPI
#include <os2.h>

ODEventType    eventType;

RemoveDispatchModule(eventType);
```

-----

## RemoveDispatchModule Parameter - eventType

**eventType** ([ODEventType](#)) - input

A platform-specific event code that specifies the type of event to be handled by the new dispatch module. On OS/2, this is a PM message ID.

-----

## RemoveDispatchModule - Return Value

None.

-----

## RemoveDispatchModule - Parameters

**eventType** ([ODEventType](#)) - input

A platform-specific event code that specifies the type of event to be handled by the new dispatch module. On OS/2, this is a PM message ID.

None.

-----

## RemoveDispatchModule - Remarks

This method is not called by part editors under normal circumstances; however, your part editor may call this method to remove a custom dispatch module.

-----

## RemoveDispatchModule - Related Methods

### Related Methods

- [ODDispatcher::AddDispatchModule](#)
- [ODDispatcher::GetDispatchModule](#)



---

# RemoveDispatchModule - Topics

**Class:**  
ODDDispatcher

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## RemoveMonitor

---

## RemoveMonitor - Syntax

This method removes the specified dispatch module, which is a monitor for the specified event type, from the dispatch module dictionary.

```
#define INCL_ODDISPATCHER
#define INCL_ODAPI
#include <os2.h>

ODEventType      eventType;
ODDDispatchModule *dispatchModule;

RemoveMonitor(eventType, dispatchModule);
```

---

## RemoveMonitor Parameter - eventType

**eventType** ([ODEventType](#)) - input

A platform-specific event code that specifies the type of event to be removed from the new dispatch module. On OS/2, this is a PM message ID.

---

## RemoveMonitor Parameter - dispatchModule

**dispatchModule** ([ODDDispatchModule \\*](#)) - input

A reference to the dispatch module to be removed.

---

## RemoveMonitor - Return Value

None.

---

## RemoveMonitor - Parameters

**eventType** ([ODEventType](#)) - input

A platform-specific event code that specifies the type of event to be removed from the new dispatch module. On OS/2, this is a PM message ID.

**dispatchModule** ([ODDispatchModule \\*](#)) - input

A reference to the dispatch module to be removed.

None.

---

## RemoveMonitor - Remarks

This method is not called by part editors under normal circumstances; however, your part editor may call this method to remove a custom dispatch module as a monitor for the specified event type.

---

## RemoveMonitor - Related Methods

### Related Methods

- [ODDispatcher::AddMonitor](#)
- 

## RemoveMonitor - Topics

### Class:

[ODDispatcher](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## ShouldExit

---

# ShouldExit - Syntax

This method is called by the document shell or container application to indicate whether the document shell should terminate.

```
#define INCL_ODDISPATCHER
#define INCL_ODAPI
#include <os2.h>

ODBoolean    rv;

rv = ShouldExit();
```

---

## ShouldExit Return Value - rv

**rv** ([ODBoolean](#)) - returns  
A flag indicating whether the document shell should terminate.

kODTrue	The document shell should terminate.
kODFalse	The document shell should not terminate.

---

## ShouldExit - Parameters

**rv** ([ODBoolean](#)) - returns  
A flag indicating whether the document shell should terminate.

kODTrue	The document shell should terminate.
kODFalse	The document shell should not terminate.

---

## ShouldExit - Remarks

The document shell and container applications call this method to see whether the dispatcher recommends that their main event loop be terminated.

---

## ShouldExit - Related Methods

### Related Methods

- [ODDispatcher::Exit](#)

---

# ShouldExit - Topics

## Class:

ODDDispatcher

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

# ODDDispatchModule

**Class Definition File:** DISPMOD.IDL

## Class Hierarchy

SOMObject

ODObject

**ODDDispatchModule**

## Description

An object of the ODDDispatchModule class is used to distribute one or more event types.

Part editors do not receive events directly from the operating system; rather, OpenDoc receives events, interprets them, and dispatches them, using the dispatcher, to the appropriate part editor. The dispatcher uses dispatch module objects to dispatch specific events to individual parts. The dispatcher uses at least one internal dispatch module to handle standard events (such as, mouse clicks, keystrokes, and menu commands) of a particular platform. Typically, you do not need to subclass ODDDispatchModule or even access the internal dispatch module directly; however, you can extend the OpenDoc dispatching system by installing additional dispatch modules.

The ODDDispatchModule class is an abstract superclass that you can subclass to create a dispatch module. You can define dispatch modules to dispatch new types of events or messages, such as the custom events generated by a data glove or pen, to your part editor. Each dispatch module is responsible for determining the frame or part to which it dispatches an event.

You can create a dispatch module object either from within a shell plug-in or from within your part's methods that are called during startup.

Your part editor can also install a dispatch module as a monitor, using the dispatcher's [AddMonitor](#) method. In this case, you can monitor the event stream without interfering with it. By registering a dispatch module as a monitor for a specified event type, OpenDoc notifies the dispatch module when an event of that type occurs. You might use a monitor in a debugging environment to monitor events and display a log of the events in a window. There can be one or more monitors for each event type.

For more information related to the dispatcher, see the class description of [ODDDispatcher](#). For more information on event handling in OpenDoc, see the chapter on user events in the *OpenDoc Programming Guide* .

## Overriding Inherited Methods

The following methods are inherited and available for use by your subclass of ODDDispatchModule.

### somInit

The somInit method initializes the instance variables in a SOM object; it is inherited from the SOMObject class.

If you subclass ODDDispatchModule, you can override this method. Your override method does not need to call its inherited method; the inherited method is automatically called for you by the SOM library.

Your override of this method should initialize the new instance variables in this dispatch module object. The SOM library calls this method when this dispatch module is created. You must not do anything in this method that could potentially fail. This limits you to operations such as setting pointer variables to null, setting numeric variables to appropriate values, and making similar assignments from constants. If you have any initialization code that can potentially fail, it must be handled in this dispatch module's subclass-specific initialization method; see also the [InitDispatchModule](#) method.

### somUninit

The somUninit method disposes of the storage created for a SOM object; it is inherited from the SOMObject class.

If you subclass `ODDispatchModule`, you can override this method. Your override method does not need to call its inherited method; the inherited method is automatically called for you by the SOM library.

Your override of this method should dispose of any storage created for this dispatch module object, including any storage related to additional instance variables initialized in this dispatch module object. The SOM library calls this method when this dispatch module is deleted; this method must not fail.

## Methods

The methods defined by the `ODDispatchModule` class include:

- [InitDispatchModule](#)
- [Dispatch](#)

## Overridden Methods

There are currently no methods overridden by the `ODDispatchModule` class.

# Dispatch

## Dispatch - Syntax

This method should dispatch the specified event to the appropriate part.

```
#define INCL_ODDISPATCHMODULE
#define INCL_ODAPI
#include <os2.h>

ODEventData    *event;
ODEventInfo    *eventInfo;
ODBoolean      rv;

rv = Dispatch(event, eventInfo);
```

## Dispatch Parameter - event

**event** ([ODEventData \\*](#)) - in/out

A platform-specific structure representing an event. On return, the fields of the structure may have been modified.

## Dispatch Parameter - eventInfo

**eventInfo** ([ODEventInfo \\*](#)) - in/out

A platform-specific structure that contains additional event information. On return, the relevant fields of the structure are filled in if the event was handled.

# Dispatch Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the event was handled by a part.

kODTrue

The event was handled by a part.

kODFalse

The event is not associated with an existing part or the event was not handled.

---

## Dispatch - Parameters

**event** ([ODEventData \\*](#)) - in/out

A platform-specific structure representing an event. On return, the fields of the structure may have been modified.

**eventInfo** ([ODEventInfo \\*](#)) - in/out

A platform-specific structure that contains additional event information. On return, the relevant fields of the structure are filled in if the event was handled.

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the event was handled by a part.

kODTrue

The event was handled by a part.

kODFalse

The event is not associated with an existing part or the event was not handled.

---

## Dispatch - Remarks

OpenDoc calls this method after it has located this dispatch module in its dispatch module dictionary. This method in turn calls the part's [HandleEvent](#) method to give the part the opportunity to handle the specified event.

---

## Dispatch - Override Policy

If you subclass [ODDispatchModule](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## Dispatch - Related Methods

### Related Methods

- [ODDispatcher::Dispatch](#)
  - [ODPart::HandleEvent](#)
-

# Dispatch - Topics

## Class:

ODDispatchModule

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Related Methods](#)

---

## InitDispatchModule

---

### InitDispatchModule - Syntax

This method initializes the dispatch module object.

```
#define INCL_ODDISPATCHMODULE
#define INCL_ODAPI
#include <os2.h>

ODSession      *session;

InitDispatchModule(session);
```

---

### InitDispatchModule Parameter - session

**session** (ODSession \*) - input

A reference to the current session object.

---

### InitDispatchModule - Return Value

None.

---

### InitDispatchModule - Parameters

**session** (ODSession \*) - input  
A reference to the current session object.

None.

---

## InitDispatchModule - Remarks

This method is not called directly to initialize this dispatch module object, but it is called by a subclass-specific initialization method. By convention, every subclass of [ODDispatchModule](#) should have a separate initialization method (for example, the `InitMyDispatchModule` method) that is called when an instance of that subclass is created. The override method may have additional parameters beyond those of the `InitDispatchModule` method. The `InitMyDispatchModule` method should call the inherited `InitDispatchModule` method at the beginning of its implementation.

If you subclass [ODDispatchModule](#), your subclass-specific initialization method, rather than its `somInit` method, should handle any initialization code that can potentially fail. For example, your initialization method may attempt to allocate memory for your dispatch module.

---

## InitDispatchModule - Override Policy

If you subclass [ODDispatchModule](#), you should not override this method.

---

## InitDispatchModule - Topics

**Class:**  
[ODDispatchModule](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)

---

## ODDocument

**Class Definition File:** DOCUMENT.IDL

**Class Hierarchy**  
SOMObject  
  ODObject  
    ODRefCountObject  
      **ODDocument**

### Description

An object of the `ODDocument` class is used to represent a document and manipulate its drafts.

The `ODDocument` class is implemented differently for different platforms and storage mechanism.

The set of related classes, [ODContainer](#), `ODDocument`, [ODDraft](#), and [ODStorageUnit](#), is called a *container suite*. Container-suite classes are



implemented as an integrated set for each platform and storage mechanism because they work intimately with one another at many levels.

An OpenDoc container can contain one or more documents. Each document, in turn, can contain one or more drafts, and each draft can contain one or more storage units. Each container, document, draft and storage unit has a unique ID.

The document shell or container application creates or accesses a document object by calling the [AcquireDocument](#) method of the appropriate container. You can obtain a reference to the document's container by calling its [GetContainer](#) method. Each document object has a unique name within its container. The [GetName](#) and [SetName](#) methods retrieve and set this name.

Because each draft corresponds to a version of its document, a document can be defined as a collection of versioned drafts. The document shell or container application creates or accesses drafts by calling the documents's [CreateDraft](#), [AcquireDraft](#), and [AcquireBaseDraft](#) methods. Other ODDocument methods copy drafts between documents and discard unwanted drafts. ODDocument is responsible for ensuring that there is only one draft object for each draft in the document.

Each draft has permissions that control access to it. Drafts are created with exclusive read/write permissions. The document shell or container application can change a draft's permissions when it calls the document's [AcquireDraft](#) and [AcquireBaseDraft](#) methods. Access to a draft is guaranteed to be exclusive only if the draft has exclusive read/write permissions.

Drafts are linearly derived in a document. The drafts of a document can be thought of as a stack; the oldest draft, called the *base draft*, is at the bottom of the stack, and the most recent draft is at the top. A given draft is said to be above an earlier draft and below a more recent draft. The stack of drafts in the document is called the document's *draft history*. Although part editors can access any draft of a document, only the most recent (topmost) draft can be modified; all earlier drafts are read only.

For more information on how ODDocument and other container-suite classes are used, see the chapters on storage and OpenDoc run-time features in the *OpenDoc Programming Guide*.

**Note:** Parts are rarely involved with the manipulation of drafts in a document. OpenDoc and the user cooperate to create and manipulate drafts.

## Methods

The methods defined by the ODDocument class include:

- [AcquireBaseDraft](#)
- [AcquireDraft](#)
- [CollapseDrafts](#)
- [CreateDraft](#)
- [Exists](#)
- [GetContainer](#)
- [GetID](#)
- [GetName](#)
- [SaveToAPrevDraft](#)
- [SetBaseDraftFromForeignDraft](#)
- [SetName](#)

## Overridden Methods

There are currently no methods overridden by the ODDocument class.

---

# AcquireBaseDraft

---

## AcquireBaseDraft - Syntax

This method is called by the document shell or container application to retrieve a reference to the base draft of this document with its permissions set as specified.

```
#define INCL_ODDOCUMENT
#define INCL_ODAPI
#include <os2.h>

ODDraftPermissions    perms;
ODDraft               *rv;
```

```
rv = AcquireBaseDraft(perms);
```

---

## AcquireBaseDraft Parameter - perms

**perms** ([ODDraftPermissions](#)) - input

The target permissions for the draft. Valid permissions are dependent on the container suite; the Bento container suite supports the following values:

kODExclusiveWrite	Exclusive read/write
kODReadOnly	Read-only

---

## AcquireBaseDraft Return Value - rv

**rv** ([ODDraft \\*](#)) - returns

A reference to the base draft of this document.

---

## AcquireBaseDraft - Parameters

**perms** ([ODDraftPermissions](#)) - input

The target permissions for the draft. Valid permissions are dependent on the container suite; the Bento container suite supports the following values:

kODExclusiveWrite	Exclusive read/write
kODReadOnly	Read-only

**rv** ([ODDraft \\*](#)) - returns

A reference to the base draft of this document.

---

## AcquireBaseDraft - Remarks

This method returns a reference to the base draft for this document with its permissions set as specified in the *perms* parameter. If this document does not already have a base draft, this method creates, initializes, stores the base draft.

The permissions specified by the *perms* parameter must be consistent with the ways in which other objects are currently accessing the base draft. The restrictions placed on the permissions depend on the container suite. With the Bento container suite, the permissions may be set to exclusive read/write ([kODExclusiveWrite](#)) if the base draft is the document's only draft and no other object already has access to the draft (either read-only or exclusive read/write). The permissions can be set to read-only ([kODReadOnly](#)) if no object has exclusive read/write access to the draft.

This method increments the reference count of the returned draft object. When the caller has finished using that draft object, it should call the draft's [Release](#) method.

---

# AcquireBaseDraft - Exception Handling

kODErrInvalidPermissions

The base draft is not accessible with the specified permissions.

## AcquireBaseDraft - Related Methods

### Related Methods

- [ODDocument::AcquireDraft](#)

## AcquireBaseDraft - Topics

### Class:

ODDocument

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

## AcquireDraft

## AcquireDraft - Syntax

This method is called by the document shell or container application to retrieve a reference to the specified draft of this document with its permissions set as specified.

```
#define INCL_ODDOCUMENT
#define INCL_ODAPI
#include <os2.h>

ODDraftPermissions    perms;
ODDraftID             id;
ODDraft               *draft;
ODPositionCode        posCode;
ODBoolean             release;
ODDraft               *rv;

rv = AcquireDraft(perms, id, draft, posCode,
                  release);
```

---

## AcquireDraft Parameter - perms

**perms** (ODDraftPermissions) - input

The target permissions for the draft. Valid permissions are dependent on the container suite; the Bento container suite supports the following values:

kODExclusiveWrite	Exclusive read/write
kODReadOnly	Read-only

---

## AcquireDraft Parameter - id

**id** (ODDraftID) - input

The ID of the desired draft or kODNULLID to use the *draft* and *posCode* parameters to identify the desired draft.

---

## AcquireDraft Parameter - draft

**draft** (ODDraft \*) - input

A reference to the draft object used with the *posCode* parameter to identify the desired draft or kODNULL to use the *id* parameter to identify the desired draft.

---

## AcquireDraft Parameter - posCode

**posCode** (ODPositionCode) - input

The position, relative to the *draft* parameter, of the desired draft in this document's draft history. This parameter can also be set to one of the following values:

kODPosSame	The desired draft is the draft object specified by the <i>draft</i> parameter.
kODPosUndefined	The <i>id</i> parameter is to be used to identify the desired draft.

---

## AcquireDraft Parameter - release

**release** (ODBoolean) - input

A flag indicating whether the specified draft should be released after the desired draft is returned. This flag is used only when the *draft* and *posCode* parameters are used to identify the desired draft.

kODTrue	The specified draft should be released after the desired draft is returned.
kODFalse	The draft should not be released.

---

## AcquireDraft Return Value - rv

**rv** (ODDraft \*) - returns  
A reference to the specified draft object.

---

## AcquireDraft - Parameters

**perms** (ODDraftPermissions) - input  
The target permissions for the draft. Valid permissions are dependent on the container suite; the Bento container suite supports the following values:

kODExclusiveWrite	Exclusive read/write
kODReadOnly	Read-only

**id** (ODDraftID) - input  
The ID of the desired draft or kODNULLID to use the *draft* and *posCode* parameters to identify the desired draft.

**draft** (ODDraft \*) - input  
A reference to the draft object used with the *posCode* parameter to identify the desired draft or kODNULL to use the *id* parameter to identify the desired draft.

**posCode** (ODPositionCode) - input  
The position, relative to the *draft* parameter, of the desired draft in this document's draft history. This parameter can also be set to one of the following values:

kODPosSame	The desired draft is the draft object specified by the <i>draft</i> parameter.
kODPosUndefined	The <i>id</i> parameter is to be used to identify the desired draft.

**release** (ODBoolean) - input  
A flag indicating whether the specified draft should be released after the desired draft is returned. This flag is used only when the *draft* and *posCode* parameters are used to identify the desired draft.

kODTrue	The specified draft should be released after the desired draft is returned.
kODFalse	The draft should not be released.

**rv** (ODDraft \*) - returns  
A reference to the specified draft object.

---

## AcquireDraft - Remarks

If the *id* parameter is not null, this method uses that parameter to identify the desired draft, and the *draft*, *posCode*, and *release* parameters are ignored.

If the *id* parameter is kODNULLID, this method uses the *draft* and *posCode* parameters to identify the desired draft. When the *posCode*

parameter is `kODPosSame`, the desired draft is the draft object specified by its *draft* parameter, and this method changes the permissions of that draft. If the *release* parameter is `kODTrue`, and if the desired draft can be returned, the draft passed in the *draft* parameter is released.

Once the desired draft is identified, this method returns a reference to that draft object with its permissions set as specified in the *perms* parameter. The specified permissions must be consistent with the ways in which other objects are currently accessing the desired draft. The restrictions placed on the permissions depend on the container suite. With the Bento container suite, the permissions can be set to exclusive read/write (`kODExclusiveWrite`) if the desired draft is the document's most recent draft and no other object already has access to the draft (either read-only or exclusive read/write). The permissions may be set to read-only (`kODReadOnly`) if no object has exclusive read/write access to the draft.

This method increments the reference count of the returned draft object. When the caller has finished using that draft object, it should call the draft's [Release](#) method.

---

## AcquireDraft - Exception Handling

`kODErrCannotChangePermissions`

The draft's permissions cannot be changed if the draft is retrieved with different permissions.

`kODErrInvalidDraftID`

The specified draft ID is invalid.

`kODErrInvalidPermissions`

The target permissions are invalid.

`kODErrRefCountNotEqualOne`

An attempt was made to change the permissions for a draft while other objects have references to it.

`kODErrUnsupportedPosCode`

The specified position code is not supported.

---

## AcquireDraft - Related Methods

### Related Methods

- [ODContainer::AcquireDocument](#)
  - [ODDocument::AcquireBaseDraft](#)
- 

## AcquireDraft - Topics

### Class:

`ODDocument`

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

## CollapseDrafts

---

# CollapseDrafts - Syntax

This method is called by the document shell or container application to remove a specified range of empty drafts from this document.

```
#define INCL_ODDOCUMENT
#define INCL_ODAPI
#include <os2.h>

ODDraft      *from;
ODDraft      *to;
ODDocument   *rv;

rv = CollapseDrafts(from, to);
```

---

## CollapseDrafts Parameter - from

**from** (ODDraft \*) - input  
A reference to the first (most recent) draft in the range.

---

## CollapseDrafts Parameter - to

**to** (ODDraft \*) - input  
A reference to the last (oldest) draft in the range or KODNULL to set the draft immediately previous to (below) the *from* draft.

---

## CollapseDrafts Return Value - rv

**rv** (ODDocument \*) - returns  
A reference to this document with the specified drafts removed.

---

## CollapseDrafts - Parameters

**from** (ODDraft \*) - input  
A reference to the first (most recent) draft in the range.

**to** (ODDraft \*) - input  
A reference to the last (oldest) draft in the range or KODNULL to set the draft immediately previous to (below) the *from* draft.

**rv** (ODDocument \*) - returns  
A reference to this document with the specified drafts removed.

---

## CollapseDrafts - Remarks

When successful, this method removes all the drafts between the *from* draft (inclusive) and the *to* draft (exclusive), and returns a reference to the amended document object. This method maintains the appropriate draft topology for this document. After successful execution, the *from* draft is no longer a valid draft object.

For this method to be successful, the following conditions must be met:

- The *from* draft itself must have a reference count of 1, meaning that only the caller has a reference to that draft object.
- The *from* draft must not be this document's base draft and must be more recent than (above) the *to* draft in this document draft history.
- All the drafts targeted for removal must be empty. A draft is empty if it contains no changes from its previous draft.
- There must be no outstanding drafts between the *from* draft (exclusive) and the *to* draft (exclusive). An *outstanding draft* has a reference count greater than 0, meaning that it is being used by some object.

Before calling this method, the document shell or container application can call the [SaveToAPrevDraft](#) method to consolidate the changes in a range of drafts.

---

## CollapseDrafts - Exception Handling

kODErrCannotCollapseDrafts

The *from* draft is not a more recent draft than the *to* draft.

kODErrNonEmptyDraft

There are nonempty drafts between the *from* draft (inclusive) and the *to* draft (exclusive).

kODErrOutstandingDraft

There are outstanding drafts between the *from* draft (exclusive) and the *to* draft (exclusive) or the *from* draft has a reference count greater than 1.

---

## CollapseDrafts - Related Methods

### Related Methods

- [ODDocument::SaveToAPrevDraft](#)
- 

## CollapseDrafts - Topics

### Class:

ODDocument

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)



---

# CreateDraft

---

## CreateDraft - Syntax

This method is called by the document shell or container application to create a new most-recent draft object in this document.

```
#define INCL_ODDOCUMENT
#define INCL_ODAPI
#include <os2.h>

ODDraft      *below;
ODBoolean    releaseBelow;
ODDraft      *rv;

rv = CreateDraft(below, releaseBelow);
```

---

## CreateDraft Parameter - below

**below** (ODDraft \*) - input  
A reference to the most-recent draft for this document.

---

## CreateDraft Parameter - releaseBelow

**releaseBelow** (ODBoolean) - input  
A flag indicating whether the specified draft should be released.

kODTrue	The specified draft should be released.
kODFalse	The specified draft should not be released.

---

## CreateDraft Return Value - rv

**rv** (ODDraft \*) - returns  
A reference to the newly created draft object with exclusive read/write permissions.

---

## CreateDraft - Parameters

**below** (ODDraft \*) - input  
A reference to the most-recent draft for this document.

**releaseBelow** (ODBoolean) - input  
A flag indicating whether the specified draft should be released.

kODTrue                   The specified draft should be released.

kODFalse                 The specified draft should not be released.

**rv** (ODDraft \*) - returns  
A reference to the newly created draft object with exclusive read/write permissions.

---

## CreateDraft - Remarks

This method creates, initializes, and returns a new draft object, which is the most recent draft of this document. (To create the base draft of the document, the document shell calls the [AcquireBaseDraft](#) method instead.)

If the *releaseBelow* parameter is kODTrue, this document removes its reference to the specified draft by calling that draft's [Release](#) method.

This method initializes the reference count of the returned draft. When the caller has finished using that draft, it should call the draft's [Release](#) method.

---

## CreateDraft - Exception Handling

kODErrInvalidBelowDraft                   The document has previous drafts, and the *below* draft is not the most recent draft of the document.

kODErrInvalidPermissions                 The *below* draft has exclusive read/write permissions, and the *releaseBelow* parameter is kODFalse; this combination is illegal because only the most-recent draft can have write permission.

---

## CreateDraft - Related Methods

### Related Methods

- [ODContainer::AcquireDocument](#)
  - [ODDocument::AcquireDraft](#)
  - [ODRefCntObject::Release](#)
- 

## CreateDraft - Topics

**Class:**  
ODDocument

Select an item:

---

# Exists

---

## Exists - Syntax

This method indicates whether the specified draft exists in this document.

```
#define INCL_ODDOCUMENT
#define INCL_ODAPI
#include <os2.h>

ODDraftID      id;
ODDraft        *draft;
ODPositionCode posCode;
ODBoolean      rv;

rv = Exists(id, draft, posCode);
```

---

## Exists Parameter - id

**id** ([ODDraftID](#)) - input

The draft ID or KODNULLID to use the *draft* and *posCode* parameters to identify the desired draft.

---

## Exists Parameter - draft

**draft** ([ODDraft \\*](#)) - input

A reference to the draft object used with the *posCode* parameter to identify the desired draft or KODNULL to use the *id* parameter to identify the desired draft.

---

## Exists Parameter - posCode

**posCode** ([ODPositionCode](#)) - input

The position, relative to the *draft* parameter, of the desired draft in this document's draft history. This parameter can also be set to one of the following values:

kODPosSame	The desired draft is the draft object specified by the <i>draft</i> parameter.
kODPosUndefined	The <i>id</i> parameter is to be used to identify the desired draft.

---

## Exists Return Value - rv

**rv** (ODBoolean) - returns  
A flag indicating whether the specified draft exists in this document.

kODTrue	The draft exists in this document.
kODFalse	The draft does not exist in this document.

---

## Exists - Parameters

**id** (ODDraftID) - input  
The draft ID or kODNULLID to use the *draft* and *posCode* parameters to identify the desired draft.

**draft** (ODDraft \*) - input  
A reference to the draft object used with the *posCode* parameter to identify the desired draft or kODNULL to use the *id* parameter to identify the desired draft.

**posCode** (ODPositionCode) - input  
The position, relative to the *draft* parameter, of the desired draft in this document's draft history. This parameter can also be set to one of the following values:

kODPosSame	The desired draft is the draft object specified by the <i>draft</i> parameter.
kODPosUndefined	The <i>id</i> parameter is to be used to identify the desired draft.

**rv** (ODBoolean) - returns  
A flag indicating whether the specified draft exists in this document.

kODTrue	The draft exists in this document.
kODFalse	The draft does not exist in this document.

---

## Exists - Remarks

If the *id* parameter is not null, this method uses that parameter to identify the desired draft and ignores the *draft* and *posCode* parameters. If the *id* parameter is kODNULLID, this method uses the *draft* and *posCode* parameters to identify the desired draft.

---

## Exists - Exception Handling

kODErrInsufficientInfoInParams

No draft was specified; the *id* parameter is KODNULLID, and the *draft* parameter is KODNULL.

kODErrUnsupportedPosCode

The specified position code is not supported.

---

## Exists - Topics

### Class:

ODDocument

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

---

## GetContainer

---

## GetContainer - Syntax

This method returns a reference to the container object that created this document.

```
#define INCL_ODDOCUMENT
#define INCL_ODAPI
#include <os2.h>
```

```
ODContainer      *rv;
```

```
rv = GetContainer();
```

---

## GetContainer Return Value - rv

**rv** (ODContainer \*) - returns

A reference to the container object that created this document.

---

## GetContainer - Parameters

**rv** (ODContainer \*) - returns

A reference to the container object that created this document.

---

## GetContainer - Remarks

This method does not increment the reference count of the returned container. For that reason, if you cache the returned container, you should call its [AcquireDocument](#) method to increment its reference count and then call its [Release](#) method when you are finished using it.

---

## GetContainer - Related Methods

### Related Methods

- [ODContainer::AcquireDocument](#)
- 

## GetContainer - Topics

### Class:

ODDocument

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## GetID

---

## GetID - Syntax

This method returns the unique ID of this document object.

```
#define INCL_ODDOCUMENT
#define INCL_ODAPI
#include <os2.h>
```

```
ODDocumentID    rv;
```

```
rv = GetID();
```

---

## GetID Return Value - rv

**rv** ([ODDocumentID](#)) - returns  
The ID of this document object.

---

## GetID - Parameters

**rv** ([ODDocumentID](#)) - returns  
The ID of this document object.

---

## GetID - Topics

**Class:**  
ODDocument

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## GetName

---

## GetName - Syntax

This method returns the name of this document.

```
#define INCL_ODDOCUMENT
#define INCL_ODAPI
#include <os2.h>

ODDocumentName    rv;

rv = GetName();
```

---

## GetName Return Value - rv

**rv** ([ODDocumentName](#)) - returns  
The name of this document or an empty sequence if this document does not have a name.

---

## GetName - Parameters

**rv** ([ODDocumentName](#)) - returns  
The name of this document or an empty sequence if this document does not have a name.

---

## GetName - Remarks

Although parts can call this method, they are not usually concerned with document names.

If this document has a name, this method returns a copy of that name; otherwise, the *text* field of the [ODDocumentName](#) structure represents an empty sequence of characters; that is, the *text* field is an [ODByteArray](#) structure whose *\_length* field contains 0 and whose *\_buffer* field contains a null pointer.

---

## GetName - Exception Handling

<a href="#">kODErrInvalidPersistentFormat</a>	Unable to read the name from persistent storage because it is not a recognized format.
---	--

---

## GetName - Related Methods

### Related Methods

- [ODDocument::SetName](#)
- 

## GetName - Topics

**Class:**  
[ODDocument](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## SaveToAPrevDraft



---

## SaveToAPrevDraft - Syntax

This method is called by the document shell or container part to consolidate the changes in the specified range of drafts.

```
#define INCL_ODDOCUMENT
#define INCL_ODAPI
#include <os2.h>

ODDraft      *from;
ODDraft      *to;

SaveToAPrevDraft (from, to);
```

---

## SaveToAPrevDraft Parameter - from

**from** (ODDraft \*) - input  
A reference to the first (most recent) draft in the range.

---

## SaveToAPrevDraft Parameter - to

**to** (ODDraft \*) - input  
A reference to the last (oldest) draft in the range.

---

## SaveToAPrevDraft - Return Value

None.

---

## SaveToAPrevDraft - Parameters

**from** (ODDraft \*) - input  
A reference to the first (most recent) draft in the range.

**to** (ODDraft \*) - input  
A reference to the last (oldest) draft in the range.

None.

---

## SaveToAPrevDraft - Remarks

If the *to* parameter is `KODNULL`, it is set to the draft immediately previous to the *from* draft. This method copies the content of all the drafts between the *from* draft (inclusive) and the *to* draft (exclusive) to the *to* draft. It then makes all the drafts from the *from* draft (inclusive) to the *to* draft (exclusive) empty. The document shell or container application can call the [CollapseDrafts](#) method to delete these empty drafts from this document.

For this method to be successful, there must be no outstanding drafts between the *from* draft (exclusive) and the *to* draft (exclusive). An *outstanding draft* has a reference count greater than 0, meaning that it is being used by some object.

---

## SaveToAPrevDraft - Exception Handling

`kODErrDraftNotExists`

Either the *from* draft does not exist, the *to* draft does not exist, or the *from* draft is not a later draft than the *to* draft.

`kODErrOutstandingDraft`

There are outstanding drafts between the *from* draft (exclusive) and the *to* draft (exclusive).

---

## SaveToAPrevDraft - Related Methods

### Related Methods

- [ODDocument::CollapseDrafts](#)
  - [ODDraft::SaveToAPrevious](#)
- 

## SaveToAPrevDraft - Topics

### Class:

`ODDocument`

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

## SetBaseDraftFromForeignDraft

---

## SetBaseDraftFromForeignDraft - Syntax

This method is called by the document shell or container application to copy a draft from another document to the base draft of this document.

```
#define INCL_ODDOCUMENT
#define INCL_ODAPI
#include <os2.h>

ODDraft      *draft;

SetBaseDraftFromForeignDraft(draft);
```

---

## SetBaseDraftFromForeignDraft Parameter - draft

**draft** (ODDraft \*) - input

A reference to the draft of some other document to be copied to the base draft of this document.

---

## SetBaseDraftFromForeignDraft - Return Value

None.

---

## SetBaseDraftFromForeignDraft - Parameters

**draft** (ODDraft \*) - input

A reference to the draft of some other document to be copied to the base draft of this document.

None.

---

## SetBaseDraftFromForeignDraft - Remarks

The document shell or container application can call this method when this document is newly created to set this document's base draft to a copy of a draft of another document.

---

## SetBaseDraftFromForeignDraft - Topics

**Class:**  
ODDocument

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

## SetName

---

## SetName - Syntax

This method is called by the document shell or container application to set the name of this document.

```
#define INCL_ODDOCUMENT
#define INCL_ODAPI
#include <os2.h>

ODDocumentName    *name;

SetName (name);
```

---

## SetName Parameter - name

**name** ([ODDocumentName \\*](#)) - input  
The new name of this document.

---

## SetName - Return Value

None.

---

## SetName - Parameters

**name** ([ODDocumentName \\*](#)) - input  
The new name of this document.

None.

---

## SetName - Related Methods

### Related Methods

- [ODDocument::GetName](#)

---

## SetName - Topics

### Class:

ODDocument

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Related Methods](#)

---

## ODDraft

**Class Definition File:** DRAFT.IDL

### Class Hierarchy

SOMObject

ODObject

ODRefCntObject

**ODDraft**

### Description

An object of the ODDraft class represents a particular version of a document object.

The ODDraft class is implemented differently for different platforms and storage mechanisms.

The set of related classes, [ODContainer](#), [ODDocument](#), ODDraft, and [ODStorageUnit](#), is called a *container suite*. Container suite classes are implemented as an integrated set for each platform and storage mechanism because they work intimately with one another at many levels.

An OpenDoc document can contain one or more drafts, and each draft can contain one or more storage units. Each document, draft, and storage unit has a unique ID. If a storage unit is used to store data for a persistent object, the storage unit's ID is also used as the ID of the persistent object.

Because each draft corresponds to a version of its document, a document can be defined as a collection of versioned drafts. The document shell or container application creates or accesses a draft of a document by calling the [CreateDraft](#), [AcquireDraft](#), and [AcquireBaseDraft](#) methods of that document.

Each draft has permissions that control access to it. Drafts are created with exclusive read/write permissions. The document shell or container application can change a draft's permissions when it calls the document's [AcquireDraft](#) and [AcquireBaseDraft](#) methods. Access to a draft is guaranteed to be exclusive only if the draft has exclusive read/write permissions. Many methods of ODDraft require a particular kind of access to the draft; for example, the [CreateFrame](#) method requires write access. Before calling one of these methods, you should call the draft's [GetPermissions](#) method and check that the draft's current permissions allow the required access.

Drafts are linearly derived in a document. The drafts of a document can be thought of as a stack; the oldest draft, called the *base draft*, is at the bottom of the stack and the most recent is at the top. Although part editors can access any draft of a document, only the most recent (topmost) draft can be modified; all earlier drafts are read-only.

The ODDraft class has a set of *CreateClass* and *AcquireClass* methods for creating and retrieving persistent objects for various classes: parts, frames, link objects, and link-source objects. For example, the [CreateFrame](#) method creates a new frame object matching the specifications you provide; the [AcquireFrame](#) method recreates the [ODFrame](#) object stored in the specified storage unit.

Many OpenDoc methods modify the content of a draft, and therefore, the draft must be notified so that it can save these changes. If this draft contains no changes since it was last saved, it is said to be *clean* ; if it contains changes, it is said to be *dirty* . For example, after a successful call to the [CreateFrame](#) method, the draft is dirty. You can call the [SetChangedFromPrev](#) method when your part's content has changed; that method marks the draft as dirty so that content changes will be saved.

You can copy or *clone* the persistent objects in a draft into a specified storage unit. All cloning must be performed within a transaction; the Bento container suite allows only one cloning transaction at any given time.

- To initiate a cloning transaction, call the draft's [BeginClone](#) method. This method returns a unique *draft key* that identifies the transaction.
- To copy a persistent object, call the draft's [Clone](#) method, passing as parameters the draft key returned by the [BeginClone](#) method, the ID of the object to be cloned, the ID of the destination storage unit, and the ID of the frame object specifying the scope of the cloning operation.

The draft's [Clone](#) method calls the [CloneInto](#) method of the object being cloned. If that object has persistent references, it clones any referenced objects within the specified scope. Objects referenced by strong persistent references are strongly cloned by recursive calls to the [Clone](#) method; objects referenced by weak persistent references are weakly cloned by calls to the [WeakClone](#) method.

- To commit the cloning transaction, call the draft's [EndClone](#) method. Until the [EndClone](#) method has successfully executed, there is no guarantee that all the objects have been copied.
- To terminate the transaction unsuccessfully, call the draft's [AbortClone](#) method. You should call this method if the cloning operation cannot be completed for any reason.

For more information on how ODDraft and other container-suite classes are used, see the chapters on storage and OpenDoc run-time features in the *OpenDoc Programming Guide* .

## Methods

The methods defined by the ODDraft class include:

- [AbortClone](#)
- [AcquireDraftProperties](#)
- [AcquireFrame](#)
- [AcquireLink](#)
- [AcquireLinkSource](#)
- [AcquirePart](#)
- [AcquirePersistentObject](#)
- [AcquireStorageUnit](#)
- [BeginClone](#)
- [ChangedFromPrev](#)
- [Clone](#)
- [CreateFrame](#)
- [CreateLinkSource](#)
- [CreateLinkSpec](#)
- [CreatePart](#)
- [CreateStorageUnit](#)
- [EndClone](#)
- [Externalize](#)
- [GetDocument](#)
- [GetID](#)
- [GetPermissions](#)
- [GetPersistentObjectID](#)
- [IsValidID](#)
- [ReleasePart](#)
- [RemoveChanges](#)
- [RemoveFrame](#)
- [RemoveFromDocument](#)
- [RemoveLink](#)
- [RemoveLinkSource](#)
- [RemovePart](#)
- [RemoveStorageUnit](#)
- [SaveToAPrevious](#)
- [SetChangedFromPrev](#)
- [WeakClone](#)

## Overridden Methods

There are currently no methods overridden by the ODDraft class.

---

# AbortClone

---

## AbortClone - Syntax

This method aborts the specified cloning transaction.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>

ODDraftKey    key;

AbortClone(key);
```

---

## AbortClone Parameter - key

**key** (ODDraftKey) - input  
The draft key of the cloning transaction to be aborted.

---

## AbortClone - Return Value

None.

---

## AbortClone - Parameters

**key** (ODDraftKey) - input  
The draft key of the cloning transaction to be aborted.

None.

---

## AbortClone - Remarks

You call this method if a cloning transaction cannot be completed for any reason. The *key* parameter must be set to the draft key value

returned by the call to the [BeginClone](#) method that started the cloning transaction.

---

## AbortClone - Exception Handling

kODErrInvalidDraftKey

The specified draft key is not the draft key for the current cloning transaction.

---

## AbortClone - Related Methods

### Related Methods

- [ODDraft::BeginClone](#)
  - [ODDraft::Clone](#)
  - [ODDraft::EndClone](#)
- 

## AbortClone - Topics

### Class:

ODDraft

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## AcquireDraftProperties

---

## AcquireDraftProperties - Syntax

This method returns a reference to the storage unit in which this draft stores its properties and data that are global to this draft.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>

ODStorageUnit      *rv;

rv = AcquireDraftProperties();
```



---

## AcquireDraftProperties Return Value - rv

**rv** (ODStorageUnit \*) - returns  
A reference to the draft-properties storage unit of this draft.

---

## AcquireDraftProperties - Parameters

**rv** (ODStorageUnit \*) - returns  
A reference to the draft-properties storage unit of this draft.

---

## AcquireDraftProperties - Remarks

The draft stores information about itself in its draft-properties storage unit. In addition, your part may create properties in this storage unit to store information related to this draft.

This method increments the reference count of the returned storage unit object. When you have finished using that storage unit object, you should call its [Release](#) method.

---

## AcquireDraftProperties - Exception Handling

kODErrNoDraftProperties	A draft-properties storage unit cannot be created.
-------------------------	--

---

## AcquireDraftProperties - Topics

**Class:**  
ODDraft

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)

---

## AcquireFrame

---

## AcquireFrame - Syntax

This method returns a reference to the frame whose data is stored in the specified storage unit.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitID    id;
ODFrame            *rv;

rv = AcquireFrame(id);
```

---

## AcquireFrame Parameter - id

**id** ([ODStorageUnitID](#)) - input  
The storage-unit ID for the target frame.

---

## AcquireFrame Return Value - rv

**rv** ([ODFrame \\*](#)) - returns  
A reference to the specified frame object.

---

## AcquireFrame - Parameters

**id** ([ODStorageUnitID](#)) - input  
The storage-unit ID for the target frame.

**rv** ([ODFrame \\*](#)) - returns  
A reference to the specified frame object.

---

## AcquireFrame - Remarks

This method increments the reference count of the returned frame object. When you have finished using that frame object, you should call its [Release](#) method.

---

## AcquireFrame - Exception Handling

kODErrCannotAcquireFrame  
kODErrInvalidID

The target frame object cannot be accessed.  
The *id* parameter is invalid.

---

## AcquireFrame - Related Methods

### Related Methods

- [ODDraft::CreateFrame](#)

---

## AcquireFrame - Topics

### Class:

ODDraft

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## AcquireLink

---

## AcquireLink - Syntax

This method returns a reference to the link object whose data is stored in the specified storage unit or that can be constructed from the given link specification.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitID    id;
ODLinkSpec         *linkSpec;
ODLink             *rv;

rv = AcquireLink(id, linkSpec);
```

---

## AcquireLink Parameter - id

**id** ([ODStorageUnitID](#)) - input  
The ID of the storage unit providing the link data or KODNULLID if the link is to be constructed.

---

## AcquireLink Parameter - linkSpec

**linkSpec** ([ODLinkSpec \\*](#)) - input  
A reference to a link-specification object from which the link is to be constructed or KODNULL if the *id* parameter is not null.

---

## AcquireLink Return Value - rv

**rv** ([ODLink \\*](#)) - returns  
A reference to the specified link object.

---

## AcquireLink - Parameters

**id** ([ODStorageUnitID](#)) - input  
The ID of the storage unit providing the link data or KODNULLID if the link is to be constructed.

**linkSpec** ([ODLinkSpec \\*](#)) - input  
A reference to a link-specification object from which the link is to be constructed or KODNULL if the *id* parameter is not null.

**rv** ([ODLink \\*](#)) - returns  
A reference to the specified link object.

---

## AcquireLink - Remarks

If your part is the destination of a link, you call this method to create a link object; you also call this method in your part's [InitPartFromStorage](#) method to recreate the link object from its stored data.

If the *id* parameter is not null, the *linkSpec* parameter is ignored. In this case, the *id* parameter must be the ID of the link object's storage unit, and this method recreates the link object from the content of the specified storage unit.

If the *id* parameter is null, the *linkSpec* parameter must be a link-specification object returned by a previous call to the [CreateLinkSpec](#) method. In this case, this method uses the information in the link-specification object to construct the link object.

This method increments the reference count of the returned link object. When you have finished using that link object, you should call its [Release](#) method.

---

## AcquireLink - Exception Handling

kODErrCannotAcquireLink

The requested link object cannot be recreated from the specified storage unit or link-specification object.

kODErrInvalidID

No link object was specified; the *id* parameter is kODNULLID, and the *linkSpec* parameter is kODNULL.

---

## AcquireLink - Related Methods

### Related Methods

- [ODDraft::CreateLinkSpec](#)
- [ODPart::InitPartFromStorage](#)

---

## AcquireLink - Topics

### Class:

[ODDraft](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

## AcquireLinkSource

---

## AcquireLinkSource - Syntax

This method returns a reference to the link-source object whose data is stored in the specified storage unit.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitID    id;
ODLinkSource       *rv;

rv = AcquireLinkSource(id);
```

---

## AcquireLinkSource Parameter - id

**id** ([ODStorageUnitID](#)) - input  
The ID of the storage unit for the target link.

---

## AcquireLinkSource Return Value - rv

**rv** ([ODLinkSource \\*](#)) - returns  
A reference to the specified link-source object.

---

## AcquireLinkSource - Parameters

**id** ([ODStorageUnitID](#)) - input  
The ID of the storage unit for the target link.

**rv** ([ODLinkSource \\*](#)) - returns  
A reference to the specified link-source object.

---

## AcquireLinkSource - Remarks

If you part is the source of a link, you can call this method in your part's [InitPartFromStorage](#) method to recreate a link-source object from its stored data.

This method increments the reference count of the returned link-source object. When you have finished using that link-source object, you should call its [Release](#) method.

---

## AcquireLinkSource - Exception Handling

<a href="#">kODErrCannotAcquireLink</a>	The companion link object does not exist and cannot be created.
<a href="#">kODErrInvalidID</a>	The <i>id</i> parameter was invalid.

---

## AcquireLinkSource - Related Methods

### Related Methods

- [ODDraft::CreateLinkSource](#)
  - [ODPart::InitPartFromStorage](#)
-

# AcquireLinkSource - Topics

## Class:

ODDraft

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

-----

## AcquirePart

-----

## AcquirePart - Syntax

This method returns a reference to the part whose data is stored in the specified storage unit.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitID    id;
ODPart              *rv;

rv = AcquirePart(id);
```

-----

## AcquirePart Parameter - id

**id** ([ODStorageUnitID](#)) - input

The storage-unit ID for the target part.

-----

## AcquirePart Return Value - rv

**rv** ([ODPart \\*](#)) - returns

A reference to the specified part object.

-----

## AcquirePart - Parameters

**id** ([ODStorageUnitID](#)) - input  
The storage-unit ID for the target part.

**rv** ([ODPart \\*](#)) - returns  
A reference to the specified part object.

---

## AcquirePart - Remarks

You call this method to create a part object from its stored data.

If a part with the indicated ID does not exist, this method calls the binding object to attempt to construct a new part. If this attempt fails, [AcquirePart](#) returns [kODNULL](#). After this method executes successfully, the return value is a fully functional part object with a reference count  $\geq 1$ .

This method increments the reference count of the returned part object. When you have finished using that part object, you should call its [Release](#) method.

---

## AcquirePart - Exception Handling

[kODErrInvalidID](#)

The *id* parameter is invalid.

---

## AcquirePart - Related Methods

### Related Methods

- [ODDraft::CreatePart](#)
- 

## AcquirePart - Topics

### Class:

[ODDraft](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## AcquirePersistentObject



---

## AcquirePersistentObject - Syntax

This method returns a reference to the part or frame with the specified scripting ID.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>

ODPersistentObjectID    objectID;
ODObjectType            *objectType;
ODPersistentObject      *rv;

rv = AcquirePersistentObject(objectID, objectType);
```

---

## AcquirePersistentObject Parameter - objectID

**objectID** ([ODPersistentObjectID](#)) - input  
The scripting ID of the desired object.

---

## AcquirePersistentObject Parameter - objectType

**objectType** ([ODObjectType](#) \*) - output  
The type of the returned object. This parameter can be set to one of the following values:

kODFrameObject      A frame.  
kODPartObject        A part.

---

## AcquirePersistentObject Return Value - rv

**rv** ([ODPersistentObject](#) \*) - returns  
A reference to the part or frame with the specified scripting ID.

---

## AcquirePersistentObject - Parameters

**objectID** ([ODPersistentObjectID](#)) - input  
The scripting ID of the desired object.

**objectType** ([ODObjectType](#) \*) - output

The type of the returned object. This parameter can be set to one of the following values:

kODFrameObject	A frame.
kODPartObject	A part.

**rv** ([ODPersistentObject](#) \*) - returns

A reference to the part or frame with the specified scripting ID.

-----

## AcquirePersistentObject - Remarks

You call this method to obtain a reference to a part or frame, given the ID that identifies it for scripting purposes. This method should be used only for scripting.

This method increments the reference count of the returned persistent object. When you have finished using that persistent object, you should call its [Release](#) method.

-----

## AcquirePersistentObject - Exception Handling

kODErrInvalidParameter

The *objectID* parameter is invalid.

kODErrInvalidPersistentObjectID

This draft has no persistent object with the specified ID, or the object is of a type that cannot be acquired.

-----

## AcquirePersistentObject - Topics

### Class:

[ODDraft](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)

-----

## AcquireStorageUnit

-----

## AcquireStorageUnit - Syntax

This method returns a reference to the storage unit with the specified ID.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitID    id;
ODStorageUnit      *rv;

rv = AcquireStorageUnit(id);
```

---

## AcquireStorageUnit Parameter - id

**id** ([ODStorageUnitID](#)) - input  
The ID of the target storage unit.

---

## AcquireStorageUnit Return Value - rv

**rv** ([ODStorageUnit \\*](#)) - returns  
A reference to the specified storage-unit object.

---

## AcquireStorageUnit - Parameters

**id** ([ODStorageUnitID](#)) - input  
The ID of the target storage unit.

**rv** ([ODStorageUnit \\*](#)) - returns  
A reference to the specified storage-unit object.

---

## AcquireStorageUnit - Remarks

You call this method to recreate a storage unit object from its stored data. The *id* parameter must be the ID of a storage unit in this draft.

This method increments the reference count of the returned storage-unit object. When you have finished using that storage-unit object, you should call its [Release](#) method.

---

## AcquireStorageUnit - Exception Handling

kODErrInvalidID

The *id* parameter is invalid.

---

# AcquireStorageUnit - Related Methods

## Related Methods

- [ODDraft::CreateStorageUnit](#)

---

# AcquireStorageUnit - Topics

## Class:

ODDraft

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

# BeginClone

---

## BeginClone - Syntax

This method initiates a cloning transaction to transfer data from this draft to the specified destination draft.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>

ODDraft      *destDraft;
ODFrame      *destFrame;
ODCloneKind  kind;
ODDraftKey   rv;

rv = BeginClone(destDraft, destFrame, kind);
```

---

## BeginClone Parameter - destDraft

**destDraft** (ODDraft \*) - input

A reference to the destination draft to which objects are to be cloned.

# BeginClone Parameter - destFrame

**destFrame** (ODFrame \*) - input

A reference to the destination frame or kODNULL if the clone operation is not a paste or a drop.

This parameter allows OpenDoc to detect an attempt to move a part into one of its embedded frames. For paste operations, this parameter should be set to the part's active frame. For drop operations, this parameter should be set to the drop target frame. In all other cases, this parameter should be kODNULL.

---

# BeginClone Parameter - kind

**kind** (ODCloneKind) - input

The kind of clone operation to be performed. This parameter can be set to one of the following values:

kODCloneCopy

Copy into the content storage unit of the clipboard object or the drag-and-drop object.

kODCloneCut

Cut into the content storage unit of the clipboard object or the drag-and-drop object.

kODCloneDropCopy

Copy at the destination of a drop.

kODCloneDropMove

Move to the destination of a drop.

kODCloneFromLink

Clone from a link.

kODClonePaste

Paste from the content storage unit of the clipboard object or the drag-and-drop object.

kODCloneToLink

Clone to a link.

---

# BeginClone Return Value - rv

**rv** (ODDraftKey) - returns

The draft key for this cloning transaction.

---

# BeginClone - Parameters

**destDraft** (ODDraft \*) - input

A reference to the destination draft to which objects are to be cloned.

**destFrame** (ODFrame \*) - input

A reference to the destination frame or kODNULL if the clone operation is not a paste or a drop.

This parameter allows OpenDoc to detect an attempt to move a part into one of its embedded frames. For paste operations, this

parameter should be set to the part's active frame. For drop operations, this parameter should be set to the drop target frame. In all other cases, this parameter should be kODNULL.

**kind** ([ODCloneKind](#)) - input

The kind of clone operation to be performed. This parameter can be set to one of the following values:

kODCloneCopy

Copy into the content storage unit of the clipboard object or the drag-and-drop object.

kODCloneCut

Cut into the content storage unit of the clipboard object or the drag-and-drop object.

kODCloneDropCopy

Copy at the destination of a drop.

kODCloneDropMove

Move to the destination of a drop.

kODCloneFromLink

Clone from a link.

kODClonePaste

Paste from the content storage unit of the clipboard object or the drag-and-drop object.

kODCloneToLink

Clone to a link.

**rv** ([ODDraftKey](#)) - returns

The draft key for this cloning transaction.

---

## BeginClone - Remarks

This method sets up a cloning transaction of the specified kind and returns a unique draft key that identifies the transaction. Other methods involved in cloning require you to supply this draft key as a parameter. The Bento container suite allows only one cloning transaction at any given time.

After calling this method, you call the draft object's [Clone](#) method to clone a particular persistent object. To commit a successful cloning transaction, call this draft object's [EndClone](#) method; before the [EndClone](#) method has successfully executed, there is no guarantee that all the objects have been copied. If the cloning operation cannot be completed for any reason, you must terminate the transaction by calling this draft object's [AbortClone](#) method.

---

## BeginClone - Exception Handling

kODErrCloningInProgress

Another cloning process is in progress.

kODErrInconsistentCloneKind

The specified clone kind is used inconsistently. For example, a paste or drop clone kind can occur only following a copy or cut operation.

kODErrInvalidCloneKind

The specified clone kind is invalid.

kODErrInvalidDestinationDraft

The specified destination draft is invalid for the specified clone kind.

---

## BeginClone - Related Methods

### Related Methods

- [ODDraft::AbortClone](#)
- [ODDraft::Clone](#)

---

## BeginClone - Topics

### Class:

[ODDraft](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

## ChangedFromPrev

---

## ChangedFromPrev - Syntax

This method is called by the document shell or container application to indicate whether this draft contains any changes from the previous draft.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>

ODBoolean    rv;

rv = ChangedFromPrev();
```

---

## ChangedFromPrev Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating whether this draft contains any changes from the previous draft.

kODTrue

This draft contains changes from the previous draft.

kODFalse

This draft does not contain changes from the previous draft.

---

## ChangedFromPrev - Parameters

**rv** ([ODBoolean](#)) - returns

A flag indicating whether this draft contains any changes from the previous draft.

kODTrue

This draft contains changes from the previous draft.

kODFalse

This draft does not contain changes from the previous draft.

-----

## ChangedFromPrev - Remarks

The document shell or container application calls this method to see whether this draft contains changes from the previous draft; if so, the draft must be notified so that it can save these changes. If this draft contains no changes since it was last saved, it is said to be *clean* and this method returns kODFalse; if it contains changes, it is said to be *dirty* and this method returns kODTrue.

You can call the [SetChangedFromPrev](#) method when your part's content has changed; this marks the draft as dirty so that content changes will be saved.

Note that a draft cannot be marked clean after it has been marked dirty; however, you can call the [RemoveChanges](#) method to remove all the changes from this draft.

-----

## ChangedFromPrev - Related Methods

### Related Methods

- [ODDraft::RemoveChanges](#)
- [ODDraft::SetChangedFromPrev](#)

-----

## ChangedFromPrev - Topics

### Class:

ODDraft

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

-----

## Clone

-----

## Clone - Syntax



This method clones the specified persistent object or storage unit.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>

ODDraftKey    key;
ODID          fromObjectID;
ODID          toObjectID;
ODID          scope;
ODID          rv;

rv = Clone(key, fromObjectID, toObjectID,
           scope);
```

---

## Clone Parameter - key

**key** (ODDraftKey) - input  
The draft key of the current cloning transaction.

---

## Clone Parameter - fromObjectID

**fromObjectID** (ODID) - input  
The ID of the persistent object or storage unit to be cloned.

---

## Clone Parameter - toObjectID

**toObjectID** (ODID) - input  
The ID of the destination persistent object or storage unit or kODNULLID to create a new storage unit in the destination draft.

---

## Clone Parameter - scope

**scope** (ODID) - input  
The ID of the frame that defines the scope of this cloning operation or kODIDAll if all referenced objects are within scope.

---

## Clone Return Value - rv

**rv** (ODID) - returns

The ID of the destination persistent object or storage unit. (The ID of a persistent object is the same as the ID of its storage unit.)

---

## Clone - Parameters

**key** ([ODDraftKey](#)) - input

The draft key of the current cloning transaction.

**fromObjectID** ([ODID](#)) - input

The ID of the persistent object or storage unit to be cloned.

**toObjectID** ([ODID](#)) - input

The ID of the destination persistent object or storage unit or KODNULLID to create a new storage unit in the destination draft.

**scope** ([ODID](#)) - input

The ID of the frame that defines the scope of this cloning operation or KODIDALL if all referenced objects are within scope.

**rv** ([ODID](#)) - returns

The ID of the destination persistent object or storage unit. (The ID of a persistent object is the same as the ID of its storage unit.)

---

## Clone - Remarks

The following summary describes how to use this method:

- You start a cloning transaction by calling this draft object's [BeginClone](#) method, which returns the draft key identifying the cloning transaction.
- Next, call this draft object's [Clone](#) method, passing as parameters the draft key returned by the [BeginClone](#) method, the ID of the object to be cloned, the ID of the destination storage unit, and the frame object specifying the scope.
- Finally, call this draft object's [EndClone](#) method, passing as a parameter the draft key returned by the [BeginClone](#) method. The [EndClone](#) method commits the cloning transaction and performs the data transfer.

The *key* parameter is the draft key of the current cloning transaction, which was returned by a call to this draft object's [BeginClone](#) method and passed to the calling object's [CloneInto](#) method.

The *fromObjectID* parameter identifies the object to be cloned. If a persistent object exist with the specified ID, that persistent object is cloned; otherwise, the storage unit with the specified ID is cloned.

The *toObjectID* parameter specifies the ID of the destination storage unit. If the *toObjectID* parameter is null, a new destination storage unit is created in the destination draft.

If the object being cloned has persistent references to other objects, the *scope* parameter determines which of the referenced objects are within the scope of this cloning operation. Typically, the *scope* parameter is the ID of a frame and only those objects embedded in that frame are within scope. In the rare case in which the *scope* parameter is KODIDALL, all referenced objects are within scope.

This method passes its *key* parameter, the destination storage unit, and its *scope* parameter to the [CloneInto](#) method of the persistent object or storage unit being cloned. If that persistent object or storage unit has persistent references, its [CloneInto](#) method clones any persistently referenced objects that are within the scope of this cloning operation. Objects referenced by strong persistent references are strongly cloned by recursive calls to the [Clone](#) method; objects referenced by weak persistent references are weakly cloned by calls to the [WeakClone](#) method.

You must not use the returned ID until the end of the current cloning transaction. Furthermore, before you try to access the persistent object or storage unit with the corresponding ID, you must call the [IsValidID](#) method to verify that the ID is still valid.

---

## Clone - Exception Handling

kODErrInvalidDraftKey

The specified draft key is not the draft key for the current cloning transaction.

kODErrInvalidID

The *toObjectID* parameter did not specify a valid destination object or storage unit.

---

## Clone - Related Methods

### Related Methods

- [ODDraft::AbortClone](#)
- [ODDraft::BeginClone](#)
- [ODDraft::EndClone](#)
- [ODDraft::IsValidID](#)
- [ODDraft::WeakClone](#)
- [ODPersistentObject::CloneInto](#)
- [ODStorageUnit::CloneInto](#)

---

## Clone - Topics

### Class:

[ODDraft](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

## CreateFrame

## CreateFrame - Syntax

This method creates a new frame in this draft.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>
```

```
ODObjectType    frameType;
ODFrame         *containingFrame;
ODShape         *frameShape;
ODCanvas        *biasCanvas;
ODPart          *part;
ODTypeToken     viewType;
ODTypeToken     presentation;
ODBoolean       isSubframe;
```

```

ODBoolean    isOverlaid;
ODFrame      *rv;

rv = CreateFrame(frameType, containingFrame,
                 frameShape, biasCanvas, part, viewType,
                 presentation, isSubframe, isOverlaid);

```

---

## CreateFrame Parameter - frameType

**frameType** (ODObjectType) - input

The type of the frame to be created. This parameter can be set to one of the following values:

- kODFrameObject  
A regular frame.
- kODNonPersistentFrameObject  
A nonpersistent frame.

---

## CreateFrame Parameter - containingFrame

**containingFrame** (ODFrame \*) - input

A reference to the containing frame of the frame to be created.

---

## CreateFrame Parameter - frameShape

**frameShape** (ODShape \*) - input

A reference to the frame shape to be created.

---

## CreateFrame Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas to whose coordinate space the new frame is biased or kODNULL to use the standard platform coordinate bias.

---

## CreateFrame Parameter - part

**part** (ODPart \*) - input

A reference to the part to be displayed in the new frame.

---

# CreateFrame Parameter - viewType

**viewType** (ODTypeToken) - input

A tokenized string representing the initial view type for the new frame.

This parameter must be the tokenized form of one of the following view-type constants. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODViewAsFrame	Frame view type.
kODViewAsLargeIcon	Large-icon (standard) view type.
kODViewAsSmallIcon	Small-icon view type.
kODViewAsThumbNail	Thumbnail view type.

---

# CreateFrame Parameter - presentation

**presentation** (ODTypeToken) - input

The initial presentation of the new frame.

---

# CreateFrame Parameter - isSubframe

**isSubframe** (ODBoolean) - input

A flag indicating whether the new frame is a subframe.

kODTrue	The new frame is a subframe.
kODFalse	The new frame is be a subframe.

---

# CreateFrame Parameter - isOverlaid

**isOverlaid** (ODBoolean) - input

A flag indicating whether the new frame should be an overlaid frame.

kODTrue	The new frame should be an overlaid frame.
kODFalse	The new frame should not be an overlaid frame.

---

# CreateFrame Return Value - rv

**rv** (ODFrame \*) - returns  
A reference to the newly created frame object.

---

## CreateFrame - Parameters

**frameType** (ODObjectType) - input

The type of the frame to be created. This parameter can be set to one of the following values:

kODFrameObject  
A regular frame.  
kODNonPersistentFrameObject  
A nonpersistent frame.

**containingFrame** (ODFrame \*) - input

A reference to the containing frame of the frame to be created.

**frameShape** (ODShape \*) - input

A reference to the frame shape to be created.

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas to whose coordinate space the new frame is biased or kODNULL to use the standard platform coordinate bias.

**part** (ODPart \*) - input

A reference to the part to be displayed in the new frame.

**viewType** (ODTypeToken) - input

A tokenized string representing the initial view type for the new frame.

This parameter must be the tokenized form of one of the following view-type constants. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODViewAsFrame  
Frame view type.  
kODViewAsLargeIcon  
Large-icon (standard) view type.  
kODViewAsSmallIcon  
Small-icon view type.  
kODViewAsThumbNail  
Thumbnail view type.

**presentation** (ODTypeToken) - input

The initial presentation of the new frame.

**isSubframe** (ODBoolean) - input

A flag indicating whether the new frame is a subframe.

kODTrue  
The new frame is a subframe.  
kODFalse  
The new frame is be a subframe.

**isOverlaid** (ODBoolean) - input

A flag indicating whether the new frame should be an overlaid frame.

kODTrue  
The new frame should be an overlaid frame.  
kODFalse  
The new frame should not be an overlaid frame.

**rv** (ODFrame \*) - returns

A reference to the newly created frame object.

---

## CreateFrame - Remarks

This method constructs and returns a frame object in this draft; the new frame has the characteristics specified by parameters. You can create a regular frame (kODFrameObject) only if this draft's current permissions provide write access.

This method initializes the reference count of the returned frame. When you have finished using that frame, you should call its [Release](#) method.

If this method executes successfully, it marks this draft as dirty.

-----

## CreateFrame - Exception Handling

kODErrCannotCreateFrame

The frame object cannot be created.

kODErrInvalidObjectType

The specified frame type is not kODFrameObject or kODNonPersistentFrameObject.

-----

## CreateFrame - Related Methods

### Related Methods

- [ODDraft::AcquireFrame](#)
- [ODSession::Tokenize](#)

-----

## CreateFrame - Topics

### Class:

ODDraft

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

-----

## CreateLinkSource

-----

## CreateLinkSource - Syntax

This method creates a new link-source object in this draft.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>

ODPart          *part;
ODLinkSource     *rv;

rv = CreateLinkSource(part);
```

---

## CreateLinkSource Parameter - part

**part** (ODPart \*) - input  
A reference to the part containing the source content of the link.

---

## CreateLinkSource Return Value - rv

**rv** (ODLinkSource \*) - returns  
A reference to the newly created link-source object.

---

## CreateLinkSource - Parameters

**part** (ODPart \*) - input  
A reference to the part containing the source content of the link.

**rv** (ODLinkSource \*) - returns  
A reference to the newly created link-source object.

---

## CreateLinkSource - Remarks

You typically call this method from your part's [CreateLink](#) method when creating the source of a link. You can call this method only if this draft's current permissions provide write access.

Because all link-source objects have a unique link object companion, this method fails if the companion object cannot be created.

This method initializes the reference count of the returned link-source object. When you have finished using this link-source object, you should call its [Release](#) method.

If this method executes successfully, it marks this draft as dirty.

---

## CreateLinkSource - Exception Handling



---

## CreateLinkSource - Related Methods

### Related Methods

- [ODDraft::AcquireLinkSource](#)
  - [ODPart::CreateLink](#)
- 

## CreateLinkSource - Topics

### Class:

[ODDraft](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## CreateLinkSpec

---

## CreateLinkSpec - Syntax

This method creates a link-specification object for the specified part.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>

ODPart      *part;
ODByteArray *data;
ODLinkSpec  *rv;

rv = CreateLinkSpec(part, data);
```

---

## CreateLinkSpec Parameter - part

**part** (ODPart \*) - input

A reference to the source part creating this link specification or `KODNULL` to create an empty link specification.

---

## CreateLinkSpec Parameter - data

**data** (`ODByteArray *`) - input

A byte array whose buffer contains arbitrary data to be placed in the link specification or `KODNULL` to create an empty link specification.

---

## CreateLinkSpec Return Value - rv

**rv** (`ODLinkSpec *`) - returns

A reference to the newly created link-specification object.

---

## CreateLinkSpec - Parameters

**part** (`ODPart *`) - input

A reference to the source part creating this link specification or `KODNULL` to create an empty link specification.

**data** (`ODByteArray *`) - input

A byte array whose buffer contains arbitrary data to be placed in the link specification or `KODNULL` to create an empty link specification.

**rv** (`ODLinkSpec *`) - returns

A reference to the newly created link-specification object.

---

## CreateLinkSpec - Remarks

For a link to be created, this method must be called by both the source part and the destination part of a data transfer operation.

- A source part that supports linking calls this method to create a link-specification object. The *part* parameter is a reference to the source part. The buffer of the *data* parameter can contain any data that the source part's [CreateLink](#) method may need for creating the specified link.

In addition to writing the source content to the clipboard or drag-and-drop object, the source part calls the [WriteLinkSpec](#) method of the returned link-specification object to write link-specification data as a signal that the source can create a link.

After calling this method, the source part is responsible for deallocating the *data* parameter and its buffer.

- The user who pastes or drops the source data to a destination part can request a link by means of the Paste As dialog box. When this happens, the destination part calls the method to create an empty link-specification object. The *part* and *data* parameters are both null. The destination part then calls the [ReadLinkSpec](#) method of the returned link-specification object to read the link-specification data. The destination part creates the link by passing the link-specification object to its draft's [AcquireLink](#) method.

The calling part should delete the link-specification object when it has finished using it.

If this method executes successfully, it marks this draft as dirty.

---

# CreateLinkSpec - Related Methods

## Related Methods

- [ODDraft::AcquireLink](#)
  - [ODLinkSpec::ReadLinkSpec](#)
  - [ODLinkSpec::WriteLinkSpec](#)
  - [ODPart::CreateLink](#)
- 

# CreateLinkSpec - Topics

## Class:

[ODDraft](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

# CreatePart

---

# CreatePart - Syntax

This method creates a new part in this draft.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>

ODType      partType;
ODEditor    optionalEditor;
ODPart      *rv;

rv = CreatePart(partType, optionalEditor);
```

---

# CreatePart Parameter - partType

**partType** ([ODType](#)) - input

The part kind for the new part or KODNULL if the part kind is to be set at a later time.

---

## CreatePart Parameter - optionalEditor

**optionalEditor** ([ODEditor](#)) - input

The part editor to be used if the normal binding fails or kODNoEditor if you choose not to specify an editor.

---

## CreatePart Return Value - rv

**rv** (ODPart \*) - returns

A reference to the newly created part object.

---

## CreatePart - Parameters

**partType** ([ODType](#)) - input

The part kind for the new part or kODNULL if the part kind is to be set at a later time.

**optionalEditor** ([ODEditor](#)) - input

The part editor to be used if the normal binding fails or kODNoEditor if you choose not to specify an editor.

**rv** (ODPart \*) - returns

A reference to the newly created part object.

---

## CreatePart - Remarks

You can call this method only if this draft's current permissions provide write access.

This method calls the new part's [InitPart](#) method to initialize the part object.

This method initializes the reference count of the returned part. When you have finished using that part, you should call its [Release](#) method.

If this method executes successfully, it marks this draft as dirty.

---

## CreatePart - Exception Handling

kODErrCannotCreatePart

The specified part object cannot be created.

---

## CreatePart - Related Methods

**Related Methods**

- [ODDraft::AcquirePart](#)
- [ODPart::InitPart](#)

---

## CreatePart - Topics

### Class:

[ODDraft](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

## CreateStorageUnit

---

## CreateStorageUnit - Syntax

This method creates a new storage unit in this draft.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>

ODStorageUnit      *rv;

rv = CreateStorageUnit();
```

---

## CreateStorageUnit Return Value - rv

**rv** (ODStorageUnit \*) - returns  
A reference to the newly created storage-unit object.

---

## CreateStorageUnit - Parameters

**rv** (ODStorageUnit \*) - returns  
A reference to the newly created storage-unit object.

---

## CreateStorageUnit - Remarks

You can call this method only if this draft's current permissions provide write access.

This method initializes the reference count of the returned storage unit. When you have finished using that storage unit, you should call its [Release](#) method.

If this method executes successfully, it marks this draft as dirty.

---

## CreateStorageUnit - Related Methods

### Related Methods

- [ODDraft::AcquireStorageUnit](#)
- 

## CreateStorageUnit - Topics

### Class:

ODDraft

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## EndClone

---

## EndClone - Syntax

This method commits the specified cloning transaction.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>
```

```
ODDraftKey    key;
```

```
EndClone(key);
```

---

## EndClone Parameter - key

**key** ([ODDraftKey](#)) - input  
The draft key for the cloning transaction to be committed.

---

## EndClone - Return Value

None.

---

## EndClone - Parameters

**key** ([ODDraftKey](#)) - input  
The draft key for the cloning transaction to be committed.

None.

---

## EndClone - Remarks

You must call this method to end a successful cloning transaction. The *key* parameter must be set to the draft returned by a call to the [BeginClone](#) method that started the cloning transaction.

If the cloning transaction cannot be completed for any reason, you should call the [AbortClone](#) method instead of this method.

After this method executes successfully, the destination draft of the cloning transaction contains copies of all the persistent objects and storage units whose [CloneInto](#) methods were called by this draft's [Clone](#) method during the transaction.

---

## EndClone - Exception Handling

kODErrInvalidDraftKey

The specified draft key is not the draft key for the current cloning transaction.

kODErrMoveIntoSelf

This clone transaction attempted to move a part into one of its embedded frames or embed one of the part's display frames into another of its display frames.

---

## EndClone - Related Methods

## Related Methods

- [ODDraft::AbortClone](#)
- [ODDraft::BeginClone](#)
- [ODDraft::Clone](#)

---

# EndClone - Topics

## Class:

ODDraft

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

---

# Externalize

---

## Externalize - Syntax

This method is called by the document shell or container application to write to storage all persistent objects and storage units of this draft object.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>
```

```
ODDraft      *rv;

rv = Externalize();
```

---

## Externalize Return Value - rv

**rv** (ODDraft \*) - returns  
A reference to this draft object.

---

## Externalize - Parameters



**rv** (ODDraft \*) - returns  
A reference to this draft object.

---

## Externalize - Remarks

This method calls the [Externalize](#) method on all the persistent objects and storage-unit objects created through this draft and then writes to persistent storage any internal structures of this draft object.

---

## Externalize - Related Methods

### Related Methods

- [ODPart::ExternalizeKinds](#)
  - [ODPersistentObject::Externalize](#)
  - [ODStorageUnit::Externalize](#)
- 

## Externalize - Topics

### Class:

ODDraft

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## GetDocument

---

## GetDocument - Syntax

This method is called by the document shell or container application to retrieve a reference to the document object that created this draft.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>
```

```
ODDocument      *rv;
```

```
rv = GetDocument();
```

---

## GetDocument Return Value - rv

**rv** (ODDocument \*) - returns  
A reference to the document object that created this draft.

---

## GetDocument - Parameters

**rv** (ODDocument \*) - returns  
A reference to the document object that created this draft.

---

## GetDocument - Remarks

This method does not increment the reference count of the returned document; the caller should not call that document's [Release](#) method.

---

## GetDocument - Topics

**Class:**  
ODDraft

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

## GetID

---

## GetID - Syntax

This method is called by the document shell or container application to retrieve the draft ID of this draft.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>
```

```
ODDraftID    rv;  
  
rv = GetID();
```

---

## GetID Return Value - rv

**rv** ([ODDraftID](#)) - returns  
The ID of this draft.

---

## GetID - Parameters

**rv** ([ODDraftID](#)) - returns  
The ID of this draft.

---

## GetID - Topics

**Class:**  
ODDraft

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## GetPermissions

---

## GetPermissions - Syntax

This method returns this draft's current permissions.

```
#define INCL_ODDRAFT  
#define INCL_ODAPI  
#include <os2.h>  
  
ODDraftPermissions    rv;  
  
rv = GetPermissions();
```

---

# GetPermissions Return Value - rv

**rv** ([ODDraftPermissions](#)) - returns

The kind of access that is currently allowed for this draft. This method returns one of the following values:

kODExclusiveWrite	Read and exclusive-write access.
kODNone	No access.
kODReadOnly	Read-only access.
kODSharedWrite	Read and shared-write access.
kODTransient	Navigation-only access.

---

## GetPermissions - Parameters

**rv** ([ODDraftPermissions](#)) - returns

The kind of access that is currently allowed for this draft. This method returns one of the following values:

kODExclusiveWrite	Read and exclusive-write access.
kODNone	No access.
kODReadOnly	Read-only access.
kODSharedWrite	Read and shared-write access.
kODTransient	Navigation-only access.

---

## GetPermissions - Remarks

You can make changes to this draft only if its current permissions allow write access (exclusive or shared). Only the most recent draft of a document can allow write access.

The Bento container suite supports only the kODReadOnly and kODExclusiveWrite draft permissions.

---

## GetPermissions - Topics

### Class:

ODDraft

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

# GetPersistentObjectID

---

## GetPersistentObjectID - Syntax

This method returns the scripting ID of the specified part or frame.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>

ODPersistentObject      *object;
ODObjectType            objectType;
ODPersistentObjectID    rv;

rv = GetPersistentObjectID(object, objectType);
```

---

## GetPersistentObjectID Parameter - object

**object** (ODPersistentObject \*) - input  
A reference to the persistent object.

---

## GetPersistentObjectID Parameter - objectType

**objectType** (ODObjectType) - input  
The type of persistent object. This parameter can be set to one of the following parameters:

kODFrameObject	A frame.
kODPartObject	A part.

---

## GetPersistentObjectID Return Value - rv

**rv** (ODPersistentObjectID) - returns

---

## GetPersistentObjectID - Parameters

**object** (ODPersistentObject \*) - input  
A reference to the persistent object.

**objectType** (ODOBJECTTYPE) - input  
The type of persistent object. This parameter can be set to one of the following parameters:

kODFrameObject      A frame.  
kODPartObject        A part.

**rv** (ODPersistentObjectID) - returns

-----

## GetPersistentObjectID - Remarks

You can call this method to obtain the ID that identifies the specified part or frame for scripting purposes.

-----

## GetPersistentObjectID - Exception Handling

kODErrInvalidPersistentObject      The specified persistent object is not valid.

-----

## GetPersistentObjectID - Related Methods

### Related Methods

- [ODDraft::AcquirePersistentObject](#)

-----

## GetPersistentObjectID - Topics

**Class:**  
[ODDraft](#)

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

-----

## IsValidID

---

## IsValidID - Syntax

This method indicates whether the specified object ID is valid.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>

ODID      id;
ODBoolean rv;

rv = IsValidID(id);
```

---

## IsValidID Parameter - id

**id** ([ODID](#)) - input  
The object ID.

---

## IsValidID Return Value - rv

**rv** ([ODBoolean](#)) - returns  
A flag indicating whether the specified object ID is valid.

kODTrue	The specified object ID is valid.
kODFalse	The specified object ID is invalid.

---

## IsValidID - Parameters

**id** ([ODID](#)) - input  
The object ID.

**rv** ([ODBoolean](#)) - returns  
A flag indicating whether the specified object ID is valid.

kODTrue	The specified object ID is valid.
kODFalse	The specified object ID is invalid.

---

# IsValidID - Remarks

You should call this method after the end of a cloning transaction and before using an ID returned by the [Clone](#) or [WeakClone](#) method during that transaction.

The *id* parameter is an ID returned by the [Clone](#) or [WeakClone](#) method during the recent cloning transaction. If the corresponding object was not strongly cloned during the transaction, the ID is now **invalid** and should not be used.

---

## IsValidID - Related Methods

### Related Methods

- [ODDraft::Clone](#)
  - [ODDraft::WeakClone](#)
- 

## IsValidID - Topics

### Class:

ODDraft

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## ReleasePart

---

## ReleasePart - Syntax

This method releases the specified part of this draft.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>

ODPart      *part;

ReleasePart(part);
```

---

## ReleasePart Parameter - part



**part** (ODPart \*) - input  
A reference to the part to be released.

---

## ReleasePart - Return Value

None.

---

## ReleasePart - Parameters

**part** (ODPart \*) - input  
A reference to the part to be released.

None.

---

## ReleasePart - Remarks

You should call this method only from your part's [Release](#) method when the part's reference count is decremented to 0.

---

## ReleasePart - Topics

**Class:**  
ODDraft

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

## RemoveChanges

---

## RemoveChanges - Syntax

This method is called by the document shell or container application to remove all changes made in the draft.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>

ODDraft      *rv;

rv = RemoveChanges();
```

---

## RemoveChanges Return Value - rv

**rv** (ODDraft \*) - returns  
A reference to this draft object with its changes removed.

---

## RemoveChanges - Parameters

**rv** (ODDraft \*) - returns  
A reference to this draft object with its changes removed.

---

## RemoveChanges - Remarks

This method can be called only if the draft's current permissions provide write access.

After this method executes successfully, this draft is empty, and a subsequent call to the [ChangedFromPrev](#) method returns kODFalse.

---

## RemoveChanges - Topics

**Class:**  
ODDraft

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

## RemoveFrame

---

# RemoveFrame - Syntax

This method removes the specified frame from this draft.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>

ODFrame      *frame;

RemoveFrame(frame);
```

---

## RemoveFrame Parameter - frame

**frame** (ODFrame \*) - input  
A reference to the frame object to be removed.

---

## RemoveFrame - Return Value

None.

---

## RemoveFrame - Parameters

**frame** (ODFrame \*) - input  
A reference to the frame object to be removed.

None.

---

## RemoveFrame - Remarks

OpenDoc calls this method internally; parts, the document shell, and container applications do not call this method.

When this method is called, the reference count of the specified frame must be 1, and this draft's current permissions must provide write access.

If this method executes successfully, it marks this draft as dirty. The specified frame is no longer a valid object, and this draft no longer contains a frame object with the corresponding ID.

---

## RemoveFrame - Exception Handling

kODErrRefCountGreaterThanZero

After the specified frame was released, its reference count was greater than 0.

kODErrRefCountNotEqualOne

The reference count of the specified frame's storage unit is not 1.

---

## RemoveFrame - Related Methods

### Related Methods

- [ODDraft::CreateFrame](#)
- [ODDraft::AcquireFrame](#)
- [ODFrame::Remove](#)

---

## RemoveFrame - Topics

### Class:

ODDraft

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## RemoveFromDocument

---

## RemoveFromDocument - Syntax

This method is called by the document shell or container application to remove this draft from its document and destroy this draft.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>
```

```
RemoveFromDocument ();
```

---

## RemoveFromDocument - Return Value

None.

---

## RemoveFromDocument - Parameters

None.

---

## RemoveFromDocument - Remarks

This method can be called only if this draft is empty, is not the base draft of its document, and has a reference count of 1.

After this method executes successfully, this draft is no longer a valid draft object.

---

## RemoveFromDocument - Topics

### Class:

ODDraft

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

---

## RemoveLink

---

## RemoveLink - Syntax

This method removes the specified link object from this draft.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>
```

```
ODLink      *link;
```

```
RemoveLink(link);
```

---

## RemoveLink Parameter - link

**link** (ODLink \*) - input  
A reference to the link object to be removed.

---

## RemoveLink - Return Value

None.

---

## RemoveLink - Parameters

**link** (ODLink \*) - input  
A reference to the link object to be removed.

None.

---

## RemoveLink - Remarks

OpenDoc calls this method internally; parts, the document shell, and container applications do not call this method.

When this method is called, the reference count of the specified link object must be 1, and this draft's current permissions must provide write access.

If this method executes successfully, it marks this draft as dirty. The specified link object is no longer a valid object, and this draft no longer contains a link object with the corresponding ID.

---

## RemoveLink - Exception Handling

kODErrRefCountGreaterThanZero

After the specified frame was released, its reference count was greater than 0.

kODErrRefCountNotEqualOne

The reference count of the specified link object's storage unit is not 1.

---

## RemoveLink - Related Methods

**Related Methods**

- [ODDraft::AcquireLink](#)

---

## RemoveLink - Topics

### Class:

ODDraft

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## RemoveLinkSource

---

## RemoveLinkSource - Syntax

This method removes the specified link-source object from this draft.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>

ODLinkSource      *link;

RemoveLinkSource(link);
```

---

## RemoveLinkSource Parameter - link

**link** (ODLinkSource \*) - input  
A reference to the link-source object to be removed.

---

## RemoveLinkSource - Return Value

None.

---

## RemoveLinkSource - Parameters

**link** (ODLinkSource \*) - input  
A reference to the link-source object to be removed.

None.

---

## RemoveLinkSource - Remarks

OpenDoc calls this method internally; parts, the document shell, and container applications do not call this method.

When this method is called, the reference count of the specified link-source object must be 1, and this draft's current permissions must provide write access.

After this method executes successfully, it marks this draft as dirty. The specified link-source is no longer a valid object, and this draft no longer contains a link-source object with the corresponding ID.

---

## RemoveLinkSource - Exception Handling

kODErrRefCountGreaterThanZero	After the specified frame was released, its reference count was greater than 0.
kODErrRefCountNotEqualOne	The reference count of the specified link-source object's storage unit is not 1.

---

## RemoveLinkSource - Related Methods

### Related Methods

- [ODDraft::AcquireLinkSource](#)
- 

## RemoveLinkSource - Topics

### Class:

ODDraft

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)



---

# RemovePart

---

## RemovePart - Syntax

This method removes the specified part from this draft.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>

ODPart      *part;

RemovePart (part);
```

---

## RemovePart Parameter - part

**part** (ODPart \*) - input  
A reference to the part object to be removed.

---

## RemovePart - Return Value

None.

---

## RemovePart - Parameters

**part** (ODPart \*) - input  
A reference to the part object to be removed.

None.

---

## RemovePart - Remarks

OpenDoc calls this method internally; parts, the document shell, and container applications do not call this method.

When this method is called, the reference count of the specified part must be 1, and this draft's current permissions must provide write access.

If this method executes successfully, it marks this draft as dirty. The specified part is no longer a valid object, and this draft no longer contains a part object with the corresponding ID.

-----

## RemovePart - Exception Handling

kODErrRefCountGreaterThanZero

After the specified frame was released, its reference count was greater than 0.

kODErrRefCountNotEqualOne

The reference count of the specified part's storage unit is not 1.

-----

## RemovePart - Related Methods

### Related Methods

- [ODDraft::AcquirePart](#)
- [ODDraft::CreatePart](#)

-----

## RemovePart - Topics

### Class:

ODDraft

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

-----

## RemoveStorageUnit

-----

## RemoveStorageUnit - Syntax

This method removes the specified storage unit from this draft.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
```

```
#include <os2.h>

ODStorageUnit      *storageUnit;

RemoveStorageUnit (storageUnit);
```

-----

## RemoveStorageUnit Parameter - storageUnit

**storageUnit** (ODStorageUnit \*) - input  
A reference to the storage-unit object to be removed.

-----

## RemoveStorageUnit - Return Value

None.

-----

## RemoveStorageUnit - Parameters

**storageUnit** (ODStorageUnit \*) - input  
A reference to the storage-unit object to be removed.

None.

-----

## RemoveStorageUnit - Remarks

OpenDoc calls this method internally; parts, the document shell, and container applications do not call this method.

When this method is called, the reference count of the specified storage unit must be 1, and this draft's current permissions must provide write access.

If this method executes successfully, it marks this draft as dirty. The specified storage unit is no longer a valid object, and this draft no longer contains a storage-unit object with the corresponding ID.

**Note:** Do not use this method to remove the storage unit associated with a persistent object (for example, a part or draft).

-----

## RemoveStorageUnit - Exception Handling

kODErrInvalidRefCount

The persistent-object reference count is not correct.

---

## RemoveStorageUnit - Related Methods

### Related Methods

- [ODDraft::AcquireStorageUnit](#)
  - [ODDraft::CreateStorageUnit](#)
  - [ODStorageUnit::Remove](#)
- 

## RemoveStorageUnit - Topics

### Class:

ODDraft

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## SaveToAPrevious

---

## SaveToAPrevious - Syntax

This method is called by the document shell or container application to copy the content of this draft to a specified previous draft of the same document.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>
```

```
ODDraft      *to;
ODDraft      *rv;
```

```
rv = SaveToAPrevious(to);
```

---

## SaveToAPrevious Parameter - to

**to** (ODDraft \*) - input

A reference to the destination draft object or KODNULL for the draft immediately previous to (below) this draft.

---

## SaveToAPrevious Return Value - rv

**rv** (ODDraft \*) - returns

A reference to this draft object.

---

## SaveToAPrevious - Parameters

**to** (ODDraft \*) - input

A reference to the destination draft object or KODNULL for the draft immediately previous to (below) this draft.

**rv** (ODDraft \*) - returns

A reference to this draft object.

---

## SaveToAPrevious - Remarks

If the *to* parameter is null, it is set to the draft immediately previous to (below) this draft. This method copies the content of all the drafts between this draft (inclusive) and the *to* draft (exclusive) to the *to* draft. It then makes all the drafts from this draft (inclusive) to the *to* draft (exclusive) empty. The document shell can call the [CollapseDrafts](#) method of this draft's document object to delete these empty drafts from the document.

For this method to be successful, this draft must have a [reference](#) count of 1, meaning that only the caller has a reference to this draft object. In addition, there must be no outstanding drafts between this draft (exclusive) and the *to* draft (exclusive). An outstanding draft has a reference count greater than 0, meaning that it is being used by some object.

Calling this method is equivalent to calling the [SaveToAPrevDraft](#) method of this draft's document, passing this draft as the *from* parameter, and passing the same *to* parameter.

---

## SaveToAPrevious - Exception Handling

KODErrOutstandingDraft

There are outstanding drafts between this draft (exclusive) and the *to* draft (exclusive) or this draft has a reference count greater than 1.

---

## SaveToAPrevious - Related Methods

### Related Methods

- [ODDocument::CollapseDrafts](#)
- [ODDocument::SaveToAPrevDraft](#)

---

## SaveToAPrevious - Topics

### Class:

ODDraft

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

## SetChangedFromPrev

---

### SetChangedFromPrev - Syntax

This method marks this draft as dirty.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>
```

```
SetChangedFromPrev();
```

---

### SetChangedFromPrev - Return Value

None.

---

### SetChangedFromPrev - Parameters

None.

---

### SetChangedFromPrev - Remarks

A draft is said to be dirty if it contains changes since it was last saved. This method lets you mark a draft as dirty when your part's content changes, so that the content changes will be saved.

This method can be called only if the draft's current permissions provide write access.

Note that a draft cannot be marked clean after it has been marked dirty; however, you can call the [RemoveChanges](#) method to remove all the changes from this draft.

---

## SetChangedFromPrev - Related Methods

### Related Methods

- [ODDraft::ChangedFromPrev](#)
  - [ODDraft::RemoveChanges](#)
- 

## SetChangedFromPrev - Topics

### Class:

ODDraft

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## WeakClone

---

## WeakClone - Syntax

This method ensures that, if the specified object is cloned, weak persistent references to it are maintained.

```
#define INCL_ODDRAFT
#define INCL_ODAPI
#include <os2.h>

ODDraftKey    key;
ODID          objectID;
ODID          toObjectID;
ODID          scope;
ODID          rv;

rv = WeakClone(key, objectID, toObjectID,
               scope);
```

---

## WeakClone Parameter - key

**key** (ODDraftKey) - input  
The draft key of the current cloning transaction.

---

## WeakClone Parameter - objectID

**objectID** (ODID) - input  
The ID of the persistent object or storage unit to be weakly cloned.

---

## WeakClone Parameter - toObjectID

**toObjectID** (ODID) - input  
The ID of the destination persistent object or storage unit or kODNULLID to create a new storage unit in this draft.

---

## WeakClone Parameter - scope

**scope** (ODID) - input  
The ID of the frame that defines the scope of this clone transaction or kODIDAll if all referenced objects are within scope.

---

## WeakClone Return Value - rv

**rv** (ODID) - returns  
The ID of the duplicated persistent object or storage unit.

---

## WeakClone - Parameters

**key** (ODDraftKey) - input  
The draft key of the current cloning transaction.

**objectID** (ODID) - input  
The ID of the persistent object or storage unit to be weakly cloned.

**toObjectID** (ODID) - input



The ID of the destination persistent object or storage unit or kODNULLID to create a new storage unit in this draft.

**scope** (ODID) - input

The ID of the frame that defines the scope of this clone transaction or kODIDAll if all referenced objects are within scope.

**rv** (ODID) - returns

The ID of the duplicated persistent object or storage unit.

-----

## WeakClone - Remarks

This method is called by the [CloneInto](#) method of persistent objects. You can call this method from your part's CloneInto method if your part has weak persistent references to other objects.

The *key* parameter is the draft key of the current cloning transaction, which was returned by a call to this draft object's [BeginClone](#) method and passed to the calling object's [CloneInto](#) method.

The *objectID* parameter is the ID of an object to be weakly cloned because the calling object has a weak persistent reference to it. The WeakClone method does not guarantee that the specified object will be copied, but if the object is copied because of an existing strong persistent reference to it, the calling object's weak persistent references will be maintained across the clone transaction.

The *toObjectID* parameter specifies the ID of the destination storage unit. If the *toObjectID* parameter is null, a new destination storage unit is created in this destination draft, if necessary.

If the object being weakly cloned has persistent references to other objects, the *scope* parameter determines which of the referenced objects are within the scope of this clone operation. Usually the *scope* parameter is the ID of a frame, and only those objects embedded in that frame are within scope. In the rare case in which the *scope* parameter is kODIDAll, all referenced objects are within scope.

You must not use the returned ID until the end of the current cloning transaction. Furthermore, before you try to access the persistent object or storage unit with the corresponding ID, you must call the [IsValidID](#) method to verify that the ID is still valid.

-----

## WeakClone - Exception Handling

kODErrInvalidID

The *toObjectID* parameter did not specify a valid destination object or storage unit.

-----

## WeakClone - Related Methods

### Related Methods

- [ODDraft::AbortClone](#)
- [ODDraft::BeginClone](#)
- [ODDraft::Clone](#)
- [ODDraft::EndClone](#)
- [ODDraft::IsValidID](#)
- [ODPersistentObject::CloneInto](#)
- [ODStorageUnit::CloneInto](#)

-----

## WeakClone - Topics

**Class:**

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

# ODDragAndDrop

**Class Definition File:** DRAGDRP.IDL

## Class Hierarchy

```

SOMObject
  ODObject
    ODBaseDragAndDrop
      ODDragAndDrop
  
```

## Description

An object of the ODDragAndDrop class provides the mechanism for dragging and dropping objects within a part, between parts, between documents, and between an OpenDoc document and a non-OpenDoc application.

The drag-and-drop facility of OpenDoc depends on system services provided by the platform. They include a service to allow data to be transferred within a process, or between processes and a system-wide mouse tracking service that can notify a process about the location and state of the mouse during a drag operation.

When a document is opened, the session object creates a single drag-and-drop object. All parts of that document share the drag-and-drop object; you can obtain a reference to it by calling the session object's [GetDragAndDrop](#) method. You must not cache this object; instead, you must call the session object's [GetDragAndDrop](#) method whenever you need the drag-and-drop object.

Data transfers requested by any part operate on the drag-and-drop object's content storage unit. You can obtain a reference to that storage unit by calling the drag-and-drop object's [GetContentStorageUnit](#) method. You must not cache that storage unit; instead, you must call the [GetContentStorageUnit](#) method whenever you need to access the storage unit.

A drag initiated by an OpenDoc part moves or copies a single item, called a *drag item*. Some items that can be dragged are a content item, a text selection, and a selected embedded frame. A drag initiated by a non-OpenDoc application may have more than one drag item. For example, a drag initiated by an application that manipulates the file system might include several files.

When your part receives a mouse-down event that occurred within an item that can be dragged, you can initiate a drag. Initiating a drag involves acquiring the drag-and-drop object from the session object, clearing the drag-and-drop object's content storage unit, copying data from the item to be dragged into the content storage unit, and starting a drag action. Once the drag is initiated, the drag-and-drop object notifies a facet when the mouse passes over it. If the mouse button is released over a facet, the facet calls its part's [Drop](#) method to notify the part of the drop. The destination part can retrieve the dragged data, using the supplied drag-item iterator. A *drag-item iterator* is an object of the [ODDragItemIterator](#) class; it allows the part to iterate through the drag items and obtain a reference to the content storage unit for each one.

For more information on drag-and-drop operations, see the chapter on data transfer in the *OpenDoc Programming Guide*. For more information on using a link specification to indicate that the source part can create a link, see the class description for [ODLinkSource](#).

## OS/2 Implementation Considerations

### Standard OS/2 Rendering Mechanisms and Formats Support

Data transfer for standard OS/2 drag-and-drop rendering mechanisms and formats (RMF) is achieved using the standard OS/2 kinds as value types of the kODPropContents property in the drag-and-drop object storage unit. The following is a list of the supported standard OS/2 RMFs and their associated kinds:

#### RMF

```

<DRM_OS2FILE,DRF_BITMAP>
<DRM_OS2FILE,DRF_DIB>
<DRM_OS2FILE,DRF_DIF>
<DRM_OS2FILE,DRF_DSPBITMAP>
<DRM_OS2FILE,DRF_METAFILE>
<DRM_OS2FILE,DRF_OEMTEXT>
<DRM_OS2FILE,DRF_OWNERDISPLAY>
<DRM_OS2FILE,DRF_PTRPICT>
<DRM_OS2FILE,DRF_RTF>
  
```

#### Standard Kind

```

kODKindOS2Bitmap
kODKindOS2DIB
kODKindOS2DIF
kODKindOS2DspBitmap
kODKindOS2Metafile
kODKindOS2OEMText
kODKindOS2OwnerDisplay
kODKindOS2PtrPict
kODKindOS2RTF
  
```

<DRM\_OS2FILE,DRF\_SYLK>  
<DRM\_OS2FILE,DRF\_TEXT>  
<DRM\_OS2FILE,DRF\_TIFF>

kODKindOS2SYLK  
kODKindOS2Text  
kODKindOS2TIFF

## OS/2 RMFs and Part Registration

Parts that support any of the above standard OS/2 RMFs should register the corresponding standard kind. Parts that support any other standard or private OS/2 rendering mechanisms and formats should register the corresponding RMF pairs as kinds. This ensures that if non-OpenDoc content of a certain type is dropped into a container part, the appropriate part editor is chosen to handle the operation.

## Methods

The methods defined by the ODDragAndDrop class include:

- [CanEmbed](#)
- [CanIncorporate](#)
- [Clear](#)
- [DispatchHandler](#)
- [GetContentStorageUnit](#)
- [GetDataFromDragManager](#)
- [GetDragAttributes](#)
- [GetDragOperation](#)
- [GetDragReference](#)
- [InitDragAndDrop](#)
- [ShowPasteAsDialog](#)
- [StartDrag](#)

## Overridden Methods

There are currently no methods overridden by the ODDragAndDrop class.

---

# CanEmbed (OS/2)

---

## CanEmbed (OS/2) - Syntax

This method queries the registration manager for a part editor capable of handling the content of the specified drag-and-drop content storage unit.

```
#define INCL_ODCLIPBOARD
#define INCL_ODAPI
#include <os2.h>

ODStorageUnit      *dropSU;
ODBoolean          rv;

rv = CanEmbed(dropSU);
```

---

# CanEmbed (OS/2) Return Value - dropSU

**dropSU** (ODStorageUnit \*) - returns

A reference to the content storage unit for the drag item.

---

# CanEmbed (OS/2) Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating whether a part editor was found.

kODTrue

A part editor was found.

kODFalse

None of the registered part editors can handle the storage unit's content.

---

## CanEmbed (OS/2) - Parameters

**dropSU** ([ODStorageUnit \\*](#)) - returns

A reference to the content storage unit for the drag item.

**rv** ([ODBoolean](#)) - returns

A flag indicating whether a part editor was found.

kODTrue

A part editor was found.

kODFalse

None of the registered part editors can handle the storage unit's content.

---

## CanEmbed (OS/2) - Remarks

A container part should call this method in its [DragEnter](#) method to determine whether it can accept the drop.

---

## CanEmbed (OS/2) - Topics

**Class:**

ODDragAndDrop

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

---

## CanIncorporate (OS/2)

---

## CanIncorporate (OS/2) - Syntax

This method indicates whether the type of data in the specified drag-and-drop content storage unit matches the specified kind.

```
#define INCL_ODCLIPBOARD
#define INCL_ODAPI
#include <os2.h>

ODStorageUnit      *dropSU;
ODType              kind;
ODBoolean           rv;

rv = CanIncorporate(dropSU, kind);
```

-----

## CanIncorporate (OS/2) Return Value - dropSU

**dropSU** (ODStorageUnit \*) - returns  
A reference to the content storage unit for the drag item.

-----

## CanIncorporate (OS/2) Return Value - kind

**kind** (ODType) - returns  
An OpenDoc part kind, a standard OS/2 kind, or a private OS/2 drag-and-drop RMF pair.

-----

## CanIncorporate (OS/2) Return Value - rv

**rv** (ODBoolean) - returns  
A flag indicating whether a match was found.

kODTrue	A match was found.
kODFalse	A match was not found.

-----

## CanIncorporate (OS/2) - Parameters

**dropSU** (ODStorageUnit \*) - returns  
A reference to the content storage unit for the drag item.

**kind** (ODType) - returns  
An OpenDoc part kind, a standard OS/2 kind, or a private OS/2 drag-and-drop RMF pair.

**rv** (ODBoolean) - returns  
A flag indicating whether a match was found.

kODTrue

kODFalse

A match was found.

A match was not found.

-----

## CanIncorporate (OS/2) - Remarks

A non-container part should call this method in its [DragEnter](#) method for the kinds that it supports to determine whether it can accept the drop.

-----

## CanIncorporate (OS/2) - Topics

### Class:

ODDragAndDrop

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

-----

## Clear

-----

## Clear - Syntax

This method clears the content storage unit for this drag-and-drop object.

```
#define INCL_ODDRAGANDDROP
#define INCL_ODAPI
#include <os2.h>
```

```
Clear();
```

-----

## Clear - Return Value

None.

-----

## Clear - Parameters

None.

---

## Clear - Remarks

When your part initiates a drag, you must call this method immediately before calling the [GetContentStorageUnit](#) method. Doing so ensures that the content storage unit is empty and ready for drag-and-drop data transfer.

---

## Clear - Exception Handling

kODErrNoDragManager

No platform-specific drag system service is available.

---

## Clear - Related Methods

### Related Methods

- [ODDragAndDrop::GetContentStorageUnit](#)
- 

## Clear - Topics

### Class:

[ODDragAndDrop](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

## DispatchHandler (OS/2)

---

## DispatchHandler (OS/2) - Syntax

This method is called by the dispatcher to forward drag-and-drop events to the drag-and-drop object for processing.

```
#define INCL_ODDRAGANDDROP
#define INCL_ODAPI
#include <os2.h>

ODEventData    *evt;
ODFacet        *targetfacet;
ODBoolean      rv;

rv = DispatchHandler(evt, targetfacet);
```

---

## DispatchHandler (OS/2) Parameter - evt

**evt** (ODEventData \*) - input  
The drag-and-drop event.

---

## DispatchHandler (OS/2) Parameter - targetfacet

**targetfacet** (ODFacet \*) - input  
The target facet for the event.

---

## DispatchHandler (OS/2) Return Value - rv

**rv** (ODBoolean) - returns  
A flag indicating whether the event was handled.

kODTrue	The event was handled.
kODFalse	None of the registered part editors can handle the storage unit's content.

---

## DispatchHandler (OS/2) - Parameters

**evt** (ODEventData \*) - input  
The drag-and-drop event.

**targetfacet** (ODFacet \*) - input  
The target facet for the event.

**rv** (ODBoolean) - returns  
A flag indicating whether the event was handled.

kODTrue	The event was handled.
---------	------------------------



kODFalse

None of the registered part editors can handle the storage unit's content.

---

## DispatchHandler (OS/2) - Remarks

Parts should not call this method.

---

## DispatchHandler (OS/2) - Topics

### Class:

ODDragAndDrop

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

---

## GetContentStorageUnit

---

## GetContentStorageUnit - Syntax

This method returns a reference to the content storage unit for this drag-and-drop object.

```
#define INCL_ODDRAGANDDROP
#define INCL_ODAPI
#include <os2.h>

ODStorageUnit      *rv;

rv = GetContentStorageUnit();
```

---

## GetContentStorageUnit Return Value - rv

rv (ODStorageUnit \*) - returns

A reference to the content storage unit.

---

## GetContentStorageUnit - Parameters

**rv** (ODStorageUnit \*) - returns  
A reference to the content storage unit.

---

## GetContentStorageUnit - Remarks

When your part initiates a drag, you should first call the [Clear](#) method, then call this method and copy data from the drag item into the returned storage unit. The drag-and-drop object handles the creation and destruction of its content storage unit, so you must neither dispose of nor release the returned storage unit.

You must not cache the returned storage unit; instead, you must call this method whenever you need to access the content storage unit.

---

## GetContentStorageUnit - Exception Handling

kODErrNoDragManager	No platform-specific drag system service is available.
---------------------	--

---

## GetContentStorageUnit - Related Methods

### Related Methods

- [ODDragAndDrop::Clear](#)
- 

## GetContentStorageUnit - Topics

**Class:**  
ODDragAndDrop

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## GetDataFromDragManager (OS/2)

---

## GetDataFromDragManager (OS/2) - Syntax

This method carries on the rendering conversation with the source of the drop to obtain the data and places the rendered data in the returned storage unit, if successful.

```
#define INCL_ODDRAGANDDROP
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitView      *theSUView;
ODStorageUnit          **renderedSU;
ODBoolean              rv;

rv = GetDataFromDragManager(theSUView, renderedSU);
```

---

## GetDataFromDragManager (OS/2) Parameter - theSUView

**theSUView** (ODStorageUnitView \*) - input

The storage unit view created by focusing the drag-item storage unit on the kODDragitem value of the kODPropContents property.

---

## GetDataFromDragManager (OS/2) Parameter - renderedSU

**renderedSU** (ODStorageUnit \*\*) - output

A pointer to the storage unit containing the rendered data; the actual storage unit is created by the drag-and-drop object.

---

## GetDataFromDragManager (OS/2) Return Value - rv

**rv** (ODBoolean) - returns

A flag indicating whether the rendering was successful.

kODTrue

The data was successfully rendered.

kODFalse

The data could not be rendered.

---

## GetDataFromDragManager (OS/2) - Parameters

**theSUView** (ODStorageUnitView \*) - input

The storage unit view created by focusing the drag-item storage unit on the kODDragitem value of the kODPropContents property.

**renderedSU** (ODStorageUnit \*\*) - output

A pointer to the storage unit containing the rendered data; the actual storage unit is created by the drag-and-drop object.

**rv** (ODBoolean) - returns

A flag indicating whether the rendering was successful.

kODTrue	The data was successfully rendered.
kODFalse	The data could not be rendered.

---

## GetDataFromDragManager (OS/2) - Remarks

This method should be called by parts in their [Drop](#) method for every drag-item storage unit referenced by the drag-item iterator. Prior to calling the method, the drag-item storage unit must be focused on kODDragitem value of kODPropContents property, and a storage-unit view must be created. Then, the GetDataFromDragManager method is called. passing in the storage-unit view and a pointer for the rendered storage unit. The part must not free the returned storage unit.

---

## GetDataFromDragManager (OS/2) - Topics

### Class:

ODDragAndDrop

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

## GetDragAttributes (OS/2)

---

## GetDragAttributes (OS/2) - Syntax

This method returns additional information about the current drag-and-drop operation.

```
#define INCL_ODDRAGANDDROP
#define INCL_ODAPI
#include <os2.h>

ODULong    rv;

rv = GetDragAttributes();
```

---

## GetDragAttributes (OS/2) Return Value - rv

**rv** ([ODULong](#)) - returns

A flag specifying the drag attributes of the current drag-and-drop operation or 0 if no drag operation is in progress.

This parameter can return any of the following flags. Some flags are relevant to a part tracking a drag event that has entered one of its

facets; others are relevant to the part that is the destination of a drop. These values can be tested for the presence of a particular flag using the bitwise AND operator (for example, the & operator in C++).

kODDDragIsInSourceFrame	The drag has not left the source frame.
kODDDragIsInSourcePart	The drag has not left the source part.
kODDDropIsInSourceFrame	The drop is occurring in the source frame of the drag.
kODDDropIsInSourcePart	The drop is occurring in the same part as the source frame of the drag.
kODDDropIsMove	The drag items are being moved.
kODDDropIsCopy	The drag items are being copied.
kODDDropIsPasteAs	The destination part should display the Paste As dialog box.

-----

## GetDragAttributes (OS/2) - Parameters

**rv** ([ODULong](#)) - returns

A flag specifying the drag attributes of the current drag-and-drop operation or 0 if no drag operation is in progress.

This parameter can return any of the following flags. Some flags are relevant to a part tracking a drag event that has entered one of its facets; others are relevant to the part that is the destination of a drop. These values can be tested for the presence of a particular flag using the bitwise AND operator (for example, the & operator in C++).

kODDDragIsInSourceFrame	The drag has not left the source frame.
kODDDragIsInSourcePart	The drag has not left the source part.
kODDDropIsInSourceFrame	The drop is occurring in the source frame of the drag.
kODDDropIsInSourcePart	The drop is occurring in the same part as the source frame of the drag.
kODDDropIsMove	The drag items are being moved.
kODDDropIsCopy	The drag items are being copied.
kODDDropIsPasteAs	The destination part should display the Paste As dialog box.

-----

## GetDragAttributes (OS/2) - Remarks

If a drag enters your part, or a drop occurs in your part, you can call this method to determine how your part should respond.

- If your part is tracking a drag that has entered one of its facets, you can decide whether to provide the user visual feedback by checking whether the kODDDragIsInSourceFrame or kODDDragIsInSourcePart flags are set in the returned value.
- If a drop occurred in your part, you can determine the appropriate response by checking which of the drop flags (kODDDropIsInSourceFrame, kODDDropIsInSourcePart, kODDDropIsMove, kODDDropIsCopy, or kODDDropIsPasteAs) are set in the returned value.

-----

## GetDragAttributes (OS/2) - Exception Handling

kODErrNoDragManager

No platform-specific drag system service is available.

---

# GetDragAttributes (OS/2) - Topics

## Class:

ODDragAndDrop

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

---

# GetDragOperation (OS/2)

---

## GetDragOperation (OS/2) - Syntax

This method returns the type of operation that the target of a drop should perform when the drop occurs.

```
#define INCL_ODDRAGANDDROP
#define INCL_ODAPI
#include <os2.h>

ODUShort    rv;

rv = GetDragOperation();
```

---

## GetDragOperation (OS/2) Return Value - rv

**rv** ([ODUShort](#)) - returns

A flag specifying the modified drag operation. This value is the one found in the *usOperation* field of the PM drag information structure ([DRAGINFO](#)). This parameter can return one of the following values:

DO_COPY	Execute a copy operation. The <b>Ctrl</b> key is pressed.
DO_DEFAULT	Execute default operation. No modifier keys are pressed.
DO_LINK	Execute a link operation. The <b>Ctrl+Shift</b> keys are pressed.
DO_MOVE	Execute a move operation. The <b>Shift</b> key is pressed.
DO_UNKNOWN	An undefined combination of keys is pressed.

---

## GetDragOperation (OS/2) - Parameters

**rv** ([ODUShort](#)) - returns

A flag specifying the modified drag operation. This value is the one found in the *usOperation* field of the PM drag information structure ([DRAGINFO](#)). This parameter can return one of the following values:

DO_COPY	Execute a copy operation. The <b>Ctrl</b> key is pressed.
DO_DEFAULT	Execute default operation. No modifier keys are pressed.
DO_LINK	Execute a link operation. The <b>Ctrl+Shift</b> keys are pressed.
DO_MOVE	Execute a move operation. The <b>Shift</b> key is pressed.
DO_UNKNOWN	An undefined combination of keys is pressed.

-----

## GetDragOperation (OS/2) - Remarks

Parts should call this method in their [DragEnter](#) or [DragWithin](#) methods to determine if the requested drop operation is acceptable and then respond appropriately.

-----

## GetDragOperation (OS/2) - Topics

### Class:

[ODDragAndDrop](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

-----

## GetDragReference (OS/2)

-----

## GetDragReference (OS/2) - Syntax

This method returns drag reference number associated with the current drag operation.

```
#define INCL_ODDRAGANDDROP
#define INCL_ODAPI
#include <os2.h>

ODPlatformDragReference    rv;

rv = GetDragReference();
```

-----

# GetDragReference (OS/2) Return Value - rv

**rv** ([ODPlatformDragReference](#)) - returns

A platform-specific drag reference number of the current drag operation or KODNULL if no drag operation is currently in progress.

---

## GetDragReference (OS/2) - Parameters

**rv** ([ODPlatformDragReference](#)) - returns

A platform-specific drag reference number of the current drag operation or KODNULL if no drag operation is currently in progress.

---

## GetDragReference (OS/2) - Remarks

You should call this method to obtain the reference number to be used in direct calls to the drag manager, for example, to perform drag highlighting.

---

## GetDragReference (OS/2) - Exception Handling

KODErrNoDragManager

No platform-specific drag system service is available.

---

## GetDragReference (OS/2) - Topics

### Class:

ODDragAndDrop

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

---

## InitDragAndDrop (OS/2)

---

## InitDragAndDrop (OS/2) - Syntax



This method provides initialization needed to perform drag-and-drop operations in OpenDoc.

```
#define INCL_ODDRAGANDDROP
#define INCL_ODAPI
#include <os2.h>

ODSession      *session;

InitDragAndDrop(session);
```

---

## InitDragAndDrop (OS/2) Parameter - session

**session** (ODSession \*) - input  
A reference to the session object.

---

## InitDragAndDrop (OS/2) - Return Value

None.

---

## InitDragAndDrop (OS/2) - Parameters

**session** (ODSession \*) - input  
A reference to the session object.

None.

---

## InitDragAndDrop (OS/2) - Remarks

This method is called during session initialization. It should not be called by part handlers.

---

## InitDragAndDrop (OS/2) - Topics

**Class:**  
ODDragAndDrop

Select an item:  
[Syntax](#)  
[Parameters](#)

---

## ShowPasteAsDialog (OS/2)

---

### ShowPasteAsDialog (OS/2) - Syntax

This method displays the Paste As dialog box and sets the appropriate dialog items according to the input parameters.

```
#define INCL_ODDRAGANDDROP
#define INCL_ODAPI
#include <os2.h>

ODBoolean          canPasteLink;
ODPasteAsMergeSetting mergeSetting;
ODFacet            *facet;
ODTypeToken        viewType;
ODStorageUnit       *contentSU;
ODPasteAsResult     *theResult;
ODBoolean          rv;

rv = ShowPasteAsDialog(canPasteLink, mergeSetting,
                      facet, viewType, contentSU, theResult);
```

---

### ShowPasteAsDialog (OS/2) Parameter - canPasteLink

**canPasteLink** (ODBoolean) - input

A flag indicating whether the destination part allows a link to be created. This parameter can be set to one of the following values:

kODTrue	Links can be created.
kODFalse	Links cannot be created.

---

### ShowPasteAsDialog (OS/2) Parameter - mergeSetting

**mergeSetting** (ODPasteAsMergeSetting) - input

A flag indicating whether embedding and merging are supported. This value determines which radio button in the **At the Destination** group box is initially selected and whether the other button is available.

kODPasteAsEmbed	<b>Embed As</b> is initially selected; <b>Merge with Contents</b> is available.
kODPasteAsEmbedOnly	<b>Embed As</b> is selected; <b>Merge with Contents</b> is disabled.
kODPasteAsMerge	<b>Merge with Contents</b> is initially selected; <b>Embed As</b> is available.
kODPasteAsMergeOnly	

---

## ShowPasteAsDialog (OS/2) Parameter - facet

**facet** (ODFacet \*) - input  
A reference to the facet in which the drop is to occur.

---

## ShowPasteAsDialog (OS/2) Parameter - viewType

**viewType** (ODTypeToken) - input  
A tokenized string representing the initial setting for the view type of the embedded part (if embedding is chosen).  
  
This parameter must be the tokenized form of one of the following view-type constants. You can call the session's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODViewAsFrame  
Framed view type.  
kODViewAsLargeIcon  
Large-icon (standard) view type  
kODViewAsSmallIcon  
Small-icon view type.  
kODViewAsThumbNail  
Thumbnail view type

---

## ShowPasteAsDialog (OS/2) Parameter - contentSU

**contentSU** (ODStorageUnit \*) - input  
A reference to the content storage unit for the drag item to be pasted.

---

## ShowPasteAsDialog (OS/2) Parameter - theResult

**theResult** (ODPasteAsResult \*) - output  
A structure reflecting the user's selection in the Paste As dialog box.

---

## ShowPasteAsDialog (OS/2) Return Value - rv

**rv** (ODBoolean) - returns  
A flag indicating whether the user clicked the **OK** button to leave the Paste As dialog box.

kODTrue

kODFalse	The user clicked the <b>OK</b> button.
	The user canceled out of the dialog box.

-----

## ShowPasteAsDialog (OS/2) - Parameters

**canPasteLink** ([ODBoolean](#)) - input

A flag indicating whether the destination part allows a link to be created. This parameter can be set to one of the following values:

kODTrue	Links can be created.
kODFalse	Links cannot be created.

**mergeSetting** ([ODPasteAsMergeSetting](#)) - input

A flag indicating whether embedding and merging are supported. This value determines which radio button in the **At the Destination** group box is initially selected and whether the other button is available.

kODPasteAsEmbed	<b>Embed As</b> is initially selected; <b>Merge with Contents</b> is available.
kODPasteAsEmbedOnly	<b>Embed As</b> is selected; <b>Merge with Contents</b> is disabled.
kODPasteAsMerge	<b>Merge with Contents</b> is initially selected; <b>Embed As</b> is available.
kODPasteAsMergeOnly	<b>Merge with Contents</b> is selected; <b>Embed As</b> is disabled.

**facet** (ODFacet \*) - input

A reference to the facet in which the drop is to occur.

**viewType** ([ODTypeToken](#)) - input

A tokenized string representing the initial setting for the view type of the embedded part (if embedding is chosen).

This parameter must be the tokenized form of one of the following view-type constants. You can call the session's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODViewAsFrame	Framed view type.
kODViewAsLargeIcon	Large-icon (standard) view type
kODViewAsSmallIcon	Small-icon view type.
kODViewAsThumbNail	Thumbnail view type

**contentSU** (ODStorageUnit \*) - input

A reference to the content storage unit for the drag item to be pasted.

**theResult** ([ODPasteAsResult](#) \*) - output

A structure reflecting the user's selection in the Paste As dialog box.

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the user clicked the **OK** button to leave the Paste As dialog box.

kODTrue	The user clicked the <b>OK</b> button.
kODFalse	The user canceled out of the dialog box.

-----

## ShowPasteAsDialog (OS/2) - Remarks

If your part is the destination of a drop and the user selects **Paste As** from the **Edit** menu, you can call this method to display the Paste As dialog box for a particular drag item. Whenever your part is the destination of a drop, you should call the [GetDragAttributes](#) method of the

drag-and-drop object; if the `kODDropsPasteAs` flag is set in the drag attributes, you must call this method.

If the `canPasteLink` parameter is `KODTrue`, the content storage unit contains a link specification and the draft permissions allow writing, then the **Paste with Link** check box is enabled and its initial setting is checked.

If the user clicks the **OK** button, this method returns `KODTrue` and sets the fields of the *theResult* output parameter to indicate the selections the user made in the Paste As dialog box; in this case, you must deallocate the non-null *selectedKind* , *translateKind* , and *editor* fields of the [ODPasteAsResult](#) structure when you are finished using them.

If the user cancels the dialog box, this method returns `KODFalse` and you do not need to take any further action.

-----

## ShowPasteAsDialog (OS/2) - Exception Handling

`kODErrIllegalNullStorageUnitInput`  
`kODErrNullFacetInput`  
`kODErrNullPasteAsResultInput`

The *contentSU* parameter is null.  
The *facet* parameter is null.  
The *theResult* parameter is null.

-----

## ShowPasteAsDialog (OS/2) - Related Methods

### Related Methods

- [ODDragAndDrop::GetDragAttributes](#)
- [ODSession::Tokenize](#)

-----

## ShowPasteAsDialog (OS/2) - Topics

### Class:

[ODDragAndDrop](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

-----

## StartDrag

-----

## StartDrag - Syntax

This method initiates a drag operation.

```
#define INCL_ODDRAGANDDROP
#define INCL_ODAPI
#include <os2.h>

ODFrame      *srcFrame;
ODType       imageType;
ODByteArray  *image;
ODPart       **destPart;
ODByteArray  *refCon;
ODDropResult rv;

rv = StartDrag(srcFrame, imageType, image,
               destPart, refCon);
```

---

## StartDrag Parameter - srcFrame

**srcFrame** (ODFrame \*) - input  
A reference to the frame in which the drag is initiated.

---

## StartDrag Parameter - imageType

**imageType** (ODType) - input  
A platform-specific drag-image type, indicating the type of image OpenDoc should display to the user as dragging feedback.

---

## StartDrag Parameter - image

**image** (ODByteArray \*) - input  
A byte array whose buffer contains the data of the image to be dragged.

---

## StartDrag Parameter - destPart

**destPart** (ODPart \*\*) - output  
A reference to the destination part. The content of this parameter is undefined if the drop is unsuccessful (the return value is kODDropFailed or kODDropUnfinished).

---

## StartDrag Parameter - refCon

**refCon** (ODByteArray \*) - input  
A byte array whose buffer contains extra platform-specific information needed for dragging.

---

## StartDrag Return Value - rv

**rv** ([ODDropResult](#)) - returns

A flag indicating the result of the drag-and-drop operation. This parameter can return any of the following values:

kODDropCopy	A successful copy operation occurred
kODDropFail	The drop operation was unsuccessful.
kODDropMove	A successful move operation occurred.
kODDropUnfinished	An asynchronous drop was started.

---

## StartDrag - Parameters

**srcFrame** (ODFrame \*) - input

A reference to the frame in which the drag is initiated.

**imageType** (ODType) - input

A platform-specific drag-image type, indicating the type of image OpenDoc should display to the user as dragging feedback.

**image** (ODByteArray \*) - input

A byte array whose buffer contains the data of the image to be dragged.

**destPart** (ODPart \*\*) - output

A reference to the destination part. The content of this parameter is undefined if the drop is unsuccessful (the return value is kODDropFailed or kODDropUnfinished).

**refCon** (ODByteArray \*) - input

A byte array whose buffer contains extra platform-specific information needed for dragging.

**rv** ([ODDropResult](#)) - returns

A flag indicating the result of the drag-and-drop operation. This parameter can return any of the following values:

kODDropCopy	A successful copy operation occurred
kODDropFail	The drop operation was unsuccessful.
kODDropMove	A successful move operation occurred.
kODDropUnfinished	An asynchronous drop was started.

---

## StartDrag - Remarks

If you part initiates a drag, you should call this method after copying data from the drag item into the content storage unit for this drag-and-drop object. If the drag item includes any frames, you should call the [SetDragging](#) method for each frame to prevent it from being dragged into itself.

The content of the *image* parameter's buffer depends on the drag-image type.

During the drag operation, the platform-specific drag manager displays the image specified in the *image* parameter and moves the image as the user moves the mouse pointer. After the user drops the image by releasing the mouse button, this method returns the result of the

operation.

On platforms that support asynchronous drag-and-drop operations, this method always returns `kODDropUnfinished`, indicating that an asynchronous operation has started.

---

## StartDrag - Exception Handling

`kODErrNoDragManager`

No platform-specific drag system service is available.

---

## StartDrag - Related Methods

### Related Methods

- [ODFrame::SetDragging](#)
- [ODPart::Drop](#)

---

## StartDrag - Topics

### Class:

`ODDragAndDrop`

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

## ODDragItemIterator

**Class Definition File:** `DGITMIT.IDL`

### Class Hierarchy

`SOMObject`

`ODObject`

`ODBaseDragItemIterator`

**`ODDragItemIterator`**

### Description

An object of the `ODDragItemIterator` class provides access to the content storage units for all the drag items of a drag-and-drop object.

Each drag item is an item being copied or moved in a drag-and-drop operation. Its content storage unit contains the data being transferred. Once a drag operation is initiated (by calling its drag-and-drop object's [StartDrag](#) method), the drag-and-drop object notifies a part when the mouse pointer passes over one of its facets. In addition to providing the mouse location, the drag-and-drop object also supplies the part with a drag-item iterator. Similarly, if the mouse pointer is released over a frame, the part is notified of the mouse location and supplied with a drag-item iterator. In both cases, the part might use the drag-item iterator to examine each drag item from the incoming drag operation and decide whether to provide destination feedback and whether to accept the drop event.



While you are using a drag-item iterator, you should not modify the list of drag items. You must postpone adding items to or removing items from the list of drag items until after you have deleted the iterator.

For more information related to drag-and-drop objects, see the description for the class [ODDragAndDrop](#). For more information on drag-and-drop operations, see the chapters on data transfer and OpenDoc run-time features in the *OpenDoc Programming Guide*. For more information on accessing objects through iterators, see the chapters on OpenDoc run-time features in the *OpenDoc Programming Guide*.

## Methods

The methods defined by the ODDragItemIterator class include:

- [First](#)
- [IsNotComplete](#)
- [Next](#)

## Overridden Methods

There are currently no methods overridden by the ODDragItemIterator class.

---

# First

---

## First - Syntax

This method begins the iteration and returns a reference to the first storage unit in the iteration sequence.

```
#define INCL_ODDRAGANDDROP
#define INCL_ODAPI
#include <os2.h>

ODStorageUnit      *rv;

rv = First();
```

---

## First Return Value - rv

**rv** (ODStorageUnit \*) - returns  
A reference to the first storage unit in the iteration sequence.

---

## First - Parameters

**rv** (ODStorageUnit \*) - returns  
A reference to the first storage unit in the iteration sequence.

---

## First - Remarks

Your part must call this method before calling this drag-item iterator's [IsNotComplete](#) method for the first time. This method may be called multiple times; each time it, resets the iteration.

Because storage units are guaranteed to be valid only as long as the iteration is [valid](#), you should never cache a reference to the returned storage unit.

This method does not increment the reference count of the returned storage unit.

---

## First - Exception Handling

kODErrIteratorOutOfSync

The list of drag items was modified while the iteration was in progress.

---

## First - Topics

### Class:

ODDragItemIterator

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

---

## IsNotComplete

---

## IsNotComplete - Syntax

This method indicates whether the iteration is incomplete.

```
#define INCL_ODDRAGANDDROP
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    rv;
```

```
rv = IsNotComplete();
```

---

## IsNotComplete Return Value - rv

**rv** ([ODBoolean](#)) - returns  
A flag indicating whether the iteration is incomplete.

kODTrue	The iteration is incomplete.
kODFalse	The iteration is complete.

-----

## IsNotComplete - Parameters

**rv** ([ODBoolean](#)) - returns  
A flag indicating whether the iteration is incomplete.

kODTrue	The iteration is incomplete.
kODFalse	The iteration is complete.

-----

## IsNotComplete - Remarks

Your part calls this method to test whether more storage units remain in the iteration sequence. This method returns kODTrue if the preceding call to the [First](#) or [Next](#) method found a storage unit. This method returns kODFalse when you have examined all the storage units (that is, when the previous call to [First](#) or [Next](#) returned kODNULL).

-----

## IsNotComplete - Exception Handling

kODErrIteratorNotInitialized	This method was called before calling either the <a href="#">First</a> or <a href="#">Next</a> method to begin the iteration.
kODErrIteratorOutOfSync	The list of drag items was modified while the iteration was in progress.

-----

## IsNotComplete - Topics

**Class:**  
[ODDragItemIterator](#)

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)

-----

## Next

---

## Next - Syntax

This method returns a reference to the next storage unit in the iteration sequence.

```
#define INCL_ODDRAGANDDROP
#define INCL_ODAPI
#include <os2.h>

ODStorageUnit      *rv;

rv = Next();
```

---

## Next Return Value - rv

**rv** (ODStorageUnit \*) - returns  
A reference to the next storage unit in the iteration sequence or KODNULL if you have reached the last storage unit.

---

## Next - Parameters

**rv** (ODStorageUnit \*) - returns  
A reference to the next storage unit in the iteration sequence or KODNULL if you have reached the last storage unit.

---

## Next - Remarks

If your part calls this method before calling this drag-item iterator's [First](#) method to begin the iteration, this method works the same as calling the [First](#) method.

Because storage units are guaranteed to be valid only as long as the iterator is valid, you should never cache a reference to the returned storage unit. This method does not increment the reference count of the returned storage unit.

---

## Next - Exception Handling

kODErrIteratorOutOfSync	The list of storage units was modified while the iteration was in progress.
-------------------------	---

---

## Next - Topics

**Class:**  
ODDragItemIterator

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)

---

# ODEmbeddedFramesIterator

**Class Definition File:** EMBFRITR.IDL

## Class Hierarchy

SOMObject  
  ODObject  
    **ODEmbeddedFramesIterator**

## Description

An object of a subclass of ODEmbeddedFramesIterator provides access to all frames directly embedded within a display frame of your part.

If your part is a container part, there is an additional class that you must subclass and implement along with your part editor (your subclass of [ODPart](#)). You must provide an embedded-frames iterator (a subclass of ODEmbeddedFramesIterator) to allow callers to access all frames directly embedded within a display frame of your part. A caller needs access to your part's embedded frames to access the parts embedded in those frames. For example, a caller might use an embedded-frames iterator if it has a spelling checker that needs to find all text parts in your document to operate.

The ODEmbeddedFramesIterator class is an abstract superclass that you must subclass to create your embedded-frames iterator. Callers create your embedded-frames iterator object by calling your part's [CreateEmbeddedFramesIterator](#) method, which returns a reference to an embedded-frames iterator object.

While you are using an embedded-frames iterator, you should not modify the list of embedded frames for the part. You must postpone adding frames to or removing frames from the list of embedded frames for the part until after you have deleted the iterator.

For more information related to frame objects, see the description of the class [ODFrame](#). For more information on accessing objects through iterators, see the chapter on OpenDoc run-time features in the *OpenDoc Programming Guide*.

## Overriding Inherited Methods

The following methods are inherited and available for use by your subclass of ODEmbeddedFramesIterator.

### somInit

The somInit method initializes the new instance variables in a SOM object; it is inherited from the SOMObject class.

If you subclass ODEmbeddedFramesIterator, you can override this method. Your override method does not need to call its inherited method; the inherited method is automatically called for you by the SOM library.

Your override of this method should initialize the instance variables in this embedded-frames iterator object. The SOM library calls this method when this embedded-frames iterator is created. You must not do anything that might fail in this method. This limits you to operations like setting pointer variables to null, setting numeric variables to appropriate values, and making similar assignments from constants. If you have any initialization code that can potentially fail, it must be handled in your embedded-frames iterator's [InitEmbeddedFramesIterator](#) method.

### somUninit

The somUninit method disposes of the storage created for a SOM object; it is inherited from the SOMObject class.

If you subclass ODEmbeddedFramesIterator, you can override this method. Your override method does not need to call its inherited method; the inherited method is automatically called for you by the SOM library.

Your override of this method should dispose of any storage created for this embedded-frames iterator object, including any storage related to additional instance variables initialized in this embedded-frames iterator object. The SOM library calls this method when this embedded-frames iterator is deleted; this method must not fail.

## Purge

The Purge method frees memory on request; it is inherited from the [ODObject](#) class.

```
ODSize Purge (in ODSize size);
```

Every subclass of [ODObject](#) can override this method and should do so if it creates caches and temporary buffers. If you subclass [ODEmbeddedFramesIterator](#), you must override this method or risk running out of available memory. Your override method must call its inherited method at some point in your implementation (it does not matter where). You should save the size value returned by the inherited method because you will need it to compute the value to return from your override method.

Your override of this method should free any caches, noncritical buffers, or objects (up to the amount of memory specified). Your override of this method should add the number of bytes actually freed to the number returned by the inherited method and return the sum as the total amount of memory released. OpenDoc calls this method in low-memory situations; you should not allocate memory for this operation.

## Methods

The methods defined by the [ODEmbeddedFramesIterator](#) class include:

- [CheckValid](#)
- [First](#)
- [InitEmbeddedFramesIterator](#)
- [IsNotComplete](#)
- [IsValid](#)
- [Next](#)
- [PartRemoved](#)

## Overridden Methods

There are currently no methods overridden by the [ODEmbeddedFramesIterator](#) class.

---

# CheckValid

---

## CheckValid - Syntax

This method should check whether this embedded-frames iterator is valid and generate an exception if it is not valid.

```
#define INCL_ODEMBEDDEDFRAMESITERATOR
#define INCL_ODAPI
#include <os2.h>
```

```
CheckValid();
```

---

## CheckValid - Return Value

None.

---

## CheckValid - Parameters

None.

---

## CheckValid - Remarks

Every subclass of [ODEEmbeddedFramesIterator](#) must test the embedded-frames iterator's validity at the beginning of the implementation of each of its noninherited methods (except for the subclass-specific initialization method) by calling either this method or the [IsValid](#) method. Unlike the [IsValid](#) method, this method has no effect if this embedded-frames iterator is valid; otherwise, it should generate an exception.

If you want to ensure that you make calls only to a valid embedded-frames iterator, without generating an exception, then call the [IsValid](#) method instead.

---

## CheckValid - Exception Handling

kODErrInvalidIterator

This embedded-frames iterator is invalid and should not be used because the part that created it no longer exists.

---

## CheckValid - Override Policy

If you subclass [ODEEmbeddedFramesIterator](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## CheckValid - Topics

### Class:

[ODEEmbeddedFramesIterator](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

---

## First

---

## First - Syntax

This method should begin the iteration and return a reference to the first frame in the iteration sequence.

```
#define INCL_ODEMBEDDEDFRAMESITERATOR
#define INCL_ODAPI
#include <os2.h>

ODFrame      *rv;

rv = First();
```

---

## First Return Value - rv

**rv** (ODFrame \*) - returns

A reference to the first frame in the iteration sequence or kODNULL if the part has no embedded frames.

---

## First - Parameters

**rv** (ODFrame \*) - returns

A reference to the first frame in the iteration sequence or kODNULL if the part has no embedded frames.

---

## First - Remarks

A client of this embedded-frames iterator calls this method before calling this embedded-frames iterator's [IsNotComplete](#) method for the first time. This method may be called multiple times; each time, it resets the iteration.

Your override of this method should not increment the reference count of the returned frame object.

---

## First - Exception Handling

kODErrIteratorOutOfSync

The list of embedded frame for the part was modified while the iteration was in progress.

---

## First - Override Policy

If you subclass [ODEmbeddedFramesIterator](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## First - Topics



**Class:**

ODEmbeddedFramesIterator

Select an item:

[Syntax](#)[Parameters](#)[Returns](#)[Remarks](#)[Override Policy](#)[Exception Handling](#)

---

## InitEmbeddedFramesIterator

---

### InitEmbeddedFramesIterator - Syntax

This method should initialize this embedded-frames iterator object.

```
#define INCL_ODEMBEDDEDFRAMESITERATOR
#define INCL_ODAPI
#include <os2.h>
```

```
ODPart      *part;
```

```
InitEmbeddedFramesIterator(part);
```

---

### InitEmbeddedFramesIterator Parameter - part

**part** (ODPart \*) - input

A reference to the part whose frames this iterator traverses.

---

### InitEmbeddedFramesIterator - Return Value

None.

---

### InitEmbeddedFramesIterator - Parameters

**part** (ODPart \*) - input

A reference to the part whose frames this iterator traverses.

None.

-----

## InitEmbeddedFramesIterator - Remarks

Your part's [CreateEmbeddedFramesIterator](#) method calls this method when this embedded-frames iterator is created. A client that uses this embedded-frames iterator does not call this method.

-----

## InitEmbeddedFramesIterator - Override Policy

If you subclass [ODEmbeddedFramesIterator](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

-----

## InitEmbeddedFramesIterator - Related Methods

### Related Methods

- [ODPart::CreateEmbeddedFramesIterator](#)

-----

## InitEmbeddedFramesIterator - Topics

### Class:

[ODEmbeddedFramesIterator](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Related Methods](#)

-----

## IsNotComplete

-----

## IsNotComplete - Syntax

This method should indicate whether the iteration is incomplete.

```

#define INCL_ODEMBEDDEDFRAMESITERATOR
#define INCL_ODAPI
#include <os2.h>

ODBoolean    rv;

rv = IsNotComplete();

```

---

## IsNotComplete Return Value - rv

**rv** (ODBoolean) - returns  
A flag indicating whether the iteration is incomplete.

kODTrue	The iteration is incomplete.
kODFalse	The iteration is complete.

---

## IsNotComplete - Parameters

**rv** (ODBoolean) - returns  
A flag indicating whether the iteration is incomplete.

kODTrue	The iteration is incomplete.
kODFalse	The iteration is complete.

---

## IsNotComplete - Remarks

A client of this embedded-frames iterator calls this method to test whether more frames remain in the iteration sequence. This method returns kODTrue if the preceding call to the [First](#) or [Next](#) method found a frame. This method returns kODFalse when you have examined all the frames (that is, when the previous call to [First](#) or [Next](#) returned kODNULL).

---

## IsNotComplete - Exception Handling

kODErrIteratorNotInitialized

This method was called before calling either the [First](#) or [Next](#) method to begin the iteration sequence.

kODErrIteratorOutOfSync

The list of embedded frames for the part was modified while the iteration was in progress.

---

## IsNotComplete - Override Policy

If you subclass [ODEmbeddedFramesIterator](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## IsNotComplete - Topics

### Class:

ODEmbeddedFramesIterator

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

---

## IsValid

---

## IsValid - Syntax

This method should indicate whether this embedded-frames iterator is valid.

```
#define INCL_ODEMBEDDEDFRAMESITERATOR
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    rv;
```

```
rv = IsValid();
```

---

## IsValid Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the embedded-frames iterator is valid.

kODTrue

The embedded-frames iterator is valid.

kODFalse

The embedded-frames iterator is not valid.

---

## IsValid - Parameters

**rv** ([ODBoolean](#)) - returns  
A flag indicating whether the embedded-frames iterator is valid.

kODTrue	The embedded-frames iterator is valid.
kODFalse	The embedded-frames iterator is not valid.

-----

## IsValid - Remarks

A client of this embedded-frames iterator calls this method if it wants to ensure that it makes calls only to a valid embedded-frames iterator, without generating an exception.

-----

## IsValid - Override Policy

If you subclass [ODEmbeddedFramesIterator](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

-----

## IsValid - Related Methods

### Related Methods

- [ODEmbeddedFramesIterator::CheckValid](#)

-----

## IsValid - Topics

**Class:**  
[ODEmbeddedFramesIterator](#)

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Related Methods](#)

-----

## Next

-----

## Next - Syntax

This method should return a reference to the next frame in the iteration sequence.

```
#define INCL_ODEMBEDDEDFRAMESITERATOR
#define INCL_ODAPI
#include <os2.h>

ODFrame      *rv;

rv = Next();
```

-----

## Next Return Value - rv

**rv** (ODFrame \*) - returns  
A reference to the next frame in the iteration sequence or kODNULL if you have reached the last frame.

-----

## Next - Parameters

**rv** (ODFrame \*) - returns  
A reference to the next frame in the iteration sequence or kODNULL if you have reached the last frame.

-----

## Next - Remarks

A client of this embedded-frames iterator calls this method. If the client calls this method before calling this embedded-frame iterator's [First](#) method to begin the iteration, this method works the same as calling the [First](#) method.

Your override of this method should not increment the reference count of the returned frame object.

-----

## Next - Exception Handling

kODErrIteratorOutOfSync

The list of embedded frames for the part was modified while the iteration was in progress.

-----

## Next - Override Policy

If you subclass [ODEmbeddedFramesIterator](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

-----

## Next - Topics

**Class:**

ODEmbeddedFramesIterator

Select an item:

[Syntax](#)[Parameters](#)[Returns](#)[Remarks](#)[Override Policy](#)[Exception Handling](#)

---

## PartRemoved

---

### PartRemoved - Syntax

This method should invalidate this embedded-frames iterator.

```
#define INCL_ODEMBEDDEDFRAMESITERATOR
#define INCL_ODAPI
#include <os2.h>
```

```
PartRemoved();
```

---

### PartRemoved - Return Value

None.

---

### PartRemoved - Parameters

None.

---

### PartRemoved - Remarks

The part whose embedded frames this iterator traverses calls this method when the part closes. This embedded-frames iterator then becomes invalid and should no longer attempt to communicate with its part; any pointers that this embedded-frames iterator had to its part should be considered invalid.

---

# PartRemoved - Override Policy

If you subclass [ODEmbeddedFramesIterator](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## PartRemoved - Topics

### Class:

[ODEmbeddedFramesIterator](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

---

## ODEExtension

**Class Definition File:** EXTENSN.IDL

### Class Hierarchy

SOMObject

  ODOObject

    ODRefCntObject

**ODEExtension**

### Description

An object of the ODEExtension class represents an extended interface to an OpenDoc object.

The OpenDoc architecture is designed to be extended. You can enhance the capabilities of and communications among your parts or other OpenDoc objects in a compound document by extending the standard OpenDoc interfaces. All subclasses of [ODOObject](#) (including shapes, facets, frames, documents, windows, and parts) can be extended. A part editor can define an extended interface for any purpose, including extensions to handle text searching, linking, specialized text formatting, database accessing, and specialized graphics processing. You can even use extensions to develop component software that goes well beyond the standard OpenDoc model of parts and compound documents.

The ODEExtension class is an abstract superclass that you can subclass to create an extended interface to a base object. For example, the [ODSemanticInterface](#) class is a subclass of ODEExtension. Callers can access an already existing extension object by calling its base object's [AcquireExtension](#) method, which returns a reference to the extension object.

The ODEExtension class itself has minimal functionality. Each extension object knows which object it is an extension of and how to release resources in itself and in its base object. Further behavior should be implemented in a subclass of ODEExtension.

### Overriding Inherited Methods

The following methods are inherited and available for use by your subclass of [ODEExtension](#).

#### somInit

The somInit method initializes the instance variables in a SOM object; it is inherited from the SOMObject class.

If you subclass ODEExtension, you can override this method. Your override method does not need to call its inherited method; the inherited method is automatically called for you by the SOM library.

Your override of this method should initialize new instance variables in this extension object. The SOM library calls this method when this extension object is created. You must not do anything that might fail in this method. This limits you to operations like setting pointer variables to null, setting numeric variables to appropriate values, and making similar assignments from constants. If you have any initialization code that can potentially fail, it must be handled in this extension's subclass-specific initialization method; see also the method [InitExtension](#).



## somUninit

The somUninit method disposes of the storage created for a SOM object; it is inherited from the SOMObject class.

If you subclass ODEExtension, you can override this method. Your override method does not need to call its inherited method; the inherited method is automatically called for you by the SOM library.

Your override method should dispose of any storage created for this extension object, including any storage related to additional instances variables initialized in this extension object. The SOM library calls this method when this extension object is deleted; this method must not fail.

## Release

The Release method decrements an object's reference count by 1; it is inherited from the [ODRefCntObject](#) class.

```
void Release ();
```

If you subclass ODEExtension, you can override this method to release an object and reclaim valuable resources, such as memory. Your override method must call its inherited method at the beginning of your implementation.

A part editor calls this method when it no longer needs a reference to this extension object. If this extension object's reference count becomes 0 and the base object is not null, the base object's [ReleaseExtension](#) method is called.

## Purge

The Purge method frees memory on request; it is inherited from the [ODOObject](#) class.

```
ODSize Purge (in ODSIZE size);
```

Every subclass of [ODOObject](#) can override this method and should do so if it creates caches and temporary buffers. If you subclass ODEExtension, you must override this method or risk running out of available memory. Your override method must call its inherited method at some point in your implementation (it does not matter where). You should save the size value returned by the inherited method because you will need it to compute the value to return from your override method.

Your override of this method should free any caches, noncritical buffers, or objects (up to the amount of memory specified). Your override of this method should add the number of bytes actually freed to the number returned by the inherited method and return the sum as the total amount of memory released. OpenDoc calls this method in low-memory situations; you should not allocate memory for this operation.

## Methods

The methods defined by the ODEExtension class include:

- [BaseRemoved](#)
- [CheckValid](#)
- [GetBase](#)
- [InitExtension](#)
- [IsValid](#)

## Overridden Methods

There are currently no methods overridden by the ODEExtension class.

---

# BaseRemoved

---

## BaseRemoved - Syntax

This method invalidates this extension object.

```
#define INCL_ODEXTENSION
#define INCL_ODAPI
#include <os2.h>
```

```
BaseRemoved();
```

-----

## BaseRemoved - Return Value

None.

-----

## BaseRemoved - Parameters

None.

-----

## BaseRemoved - Remarks

An extension object becomes invalid when its base object is removed. This extension object should no longer attempt to communicate with its base object; any pointers that this extension object had to its base object should be considered invalid. The base object of this extension should call this method from its [ReleaseAll](#) method.

-----

## BaseRemoved - Override Policy

If you subclass [ODEExtension](#), you can override this method. Your override of this method must call its inherited method at some point in your implementation (it does not matter where); you must not access the base object after calling the inherited object.

-----

## BaseRemoved - Topics

### Class:

[ODEExtension](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

-----

## CheckValid

---

## CheckValid - Syntax

This method checks whether this extension object is valid and generates an exception if it is not valid.

```
#define INCL_ODEXTENSION
#define INCL_ODAPI
#include <os2.h>
```

```
CheckValid();
```

---

## CheckValid - Return Value

None.

---

## CheckValid - Parameters

None.

---

## CheckValid - Remarks

Every subclass of [ODEExtension](#) must test the extension object's validity at the beginning of the implementation of each of its noninherited methods (except for the subclass-specific initialization method) by calling either this method or the [IsValid](#) method. Unlike the [IsValid](#) method, calling this method has no effect if this extension object is valid; otherwise, it generates an exception.

If you want to ensure that you make calls only to a valid extension object, without generating an exception, then call the [IsValid](#) method instead.

---

## CheckValid - Exception Handling

KODerrInvalidExtension

This extension object is invalid and should not be used because its base object no longer exists.

---

## CheckValid - Related Methods

## Related Methods

- [ODEExtension::IsValid](#)

---

# CheckValid - Topics

## Class:

ODEExtension

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

# GetBase

---

## GetBase - Syntax

This method returns a reference to this extension's base object.

```
#define INCL_ODEXTENSION
#define INCL_ODAPI
#include <os2.h>
```

```
ODObject      *rv;
```

```
rv = GetBase();
```

---

## GetBase Return Value - rv

**rv** (ODObject \*) - returns

A reference to this extension's base object.

---

## GetBase - Parameters

**rv** (ODObject \*) - returns  
A reference to this extension's base object.

---

## GetBase - Remarks

A client of this extension object calls this method if it has a reference to this extension object but did not save a reference to its base object.

---

## GetBase - Topics

**Class:**  
ODExtension

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

## InitExtension

---

## InitExtension - Syntax

This method initializes this extension object.

```
#define INCL_ODEXTENSION
#define INCL_ODAPI
#include <os2.h>
```

```
ODObject      *base;

InitExtension(base);
```

---

## InitExtension Parameter - base

**base** (ODObject \*) - input  
A reference to this extension's base object.

---

## InitExtension - Return Value

None.

---

## InitExtension - Parameters

**base** (ODOObject \*) - input  
A reference to this extension's base object.

None.

---

## InitExtension - Remarks

This method is not called directly to initialize this extension object but is called by a subclass-specific initialization method. By convention, every subclass of [ODEExtension](#) should have a separate initialization method (for example, the InitMyExtension method) that is called when an instance of that subclass is created. The override method may have additional parameters beyond those of the InitExtension method. The InitMyExtension method should call the inherited InitExtension method at the beginning of its implementation.

If you subclass [ODEExtension](#), your subclass-specific initialization method, rather than its somInit method, should handle any initialization code that can potentially fail. For example, your initialization method may attempt to allocate memory for your extension.

---

## InitExtension - Override Policy

If you subclass [ODEExtension](#), you should not override this method.

---

## InitExtension - Topics

**Class:**  
[ODEExtension](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)

---

## IsValid

---

# IsValid - Syntax

This method indicates whether this extension object is valid.

```
#define INCL_ODEXTENSION
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    rv;
```

```
rv = IsValid();
```

---

## IsValid Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating whether this extension object is valid.

TRUE

This extension object is valid.

FALSE

This extension object is invalid.

---

## IsValid - Parameters

**rv** ([ODBoolean](#)) - returns

A flag indicating whether this extension object is valid.

TRUE

This extension object is valid.

FALSE

This extension object is invalid.

---

## IsValid - Remarks

A client of this extension object calls this method if it wants to ensure that it makes calls only to a valid extension object, without generating an exception.

---

## IsValid - Related Methods

### Related Methods

- [ODExtension::CheckValid](#)
-

# IsValid - Topics

## Class:

ODEExtension

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

## ODFacet

**Class Definition File:** FACET.IDL

### Class Hierarchy

SOMObject

  ODObject

    ODBaseFacet

**ODFacet**

### Description

An object of the ODFacet class holds nonpersistent information about the layout of parts and describes the location of a frame on a particular canvas for display and event dispatching.

A facet object represents a visible, drawable frame or portion of a frame. In the simplest case, each embedded frame in a document is displayed and manipulated using only a single facet. In some cases, however, your part might need to display portions of the same embedded frame in several places, such as in a split view. In that case, one or more facets might manage the layout of one frame. All facets of a frame might display that frame's content identically, or a part might store user-defined part information data in the facet to distinguish among its different facets. The part information might also consist of graphics-system-specific information used for displaying the facet.

Facets are organized hierarchically. Each window or printed page has a single facet, the *root facet*, which is the topmost facet visible in the document window. All other facets contained in that window descend from the root facet. Each facet of a given frame is contained within a facet of that frame's containing frame.

On OS/2, facets that are contained within a root facet with an onscreen, dynamic canvas are each associated with a facet window. A facet window is a Presentation Manager (PM) window that is created and managed by OpenDoc. Generally, parts should not attempt to manipulate their facet windows; however, the handle for the facet window is available through the facet's [GetFacetHWND](#) method so that parts can specify the facet window as the parent and owner of embedded PM controls, such as buttons and scroll bars. Notification messages from these controls are sent to the part by the facet window in the part's [HandleEvent](#) method. The *hwnd* field in the [ODEventData](#) structure identifies these messages as having come from the facet window rather than through the OpenDoc dispatcher.

Your part creates a facet object for each visible embedded frame by calling its own display facet's [CreateEmbeddedFacet](#) method. Your part can also create a root facet (for printing) by calling the window-state object's [CreateFacet](#) method. These methods return a reference to a facet object.

Facets may or may not exist for frames that have been previously visible and that have scrolled out of view, and parts may or may not create facets for frames that are expected to be visible soon. The only requirement is that facets must exist for all currently visible frames. A visible facet always has a frame, and that frame is defined for the lifetime of the facet; once set, it cannot be changed. Facets that are not currently visible in a window may be purged from memory under low-memory conditions.

### Facet Geometry

Facets hold information regarding the geometry of their corresponding frames. Each facet maintains an active shape, a clip shape and an external transform. The clip shape and external transform must always be valid.

- The *active shape* defines the area within a frame in which the facet's embedded part is willing to receive geometry-based user events, such as mouse clicks. It is commonly identical to the frame shape of the facet's frame, but it might be modified to coincide with the used shape or some other shape. The facet's embedded part controls the active shape.
- The *clip shape* defines the area within a frame in which drawing can occur; it is the area unobscured by overlapping content of the containing part. If it is unobscured, the clip shape is identical to the frame shape of the facet's frame. The facet's containing part controls the clip shape.



- The *external transform* describes the transform that is applied to a facet to position, scale, or otherwise transform the image drawn within the facet in the coordinate space of its containing part. The facet's containing part controls the external transform.

## Facets and Canvases

A facet might possess its own canvas. If a particular facet in a window's facet hierarchy has an attached canvas, it and all of its embedded facets (and their embedded facets, and so on) draw to that canvas. Each facet inherits its canvas from its containing facet; the inherited canvas is called the *parent canvas*. For most drawing, only a window's root facet needs a canvas. For offscreen double buffering or image manipulation, however, a part can create a canvas and attach it to an embedded facet, copying the image of the offscreen canvas to its parent canvas during updates.

Because a facet can simultaneously have both a window canvas and an offscreen canvas, there are two environments to consider. The aggregate clip shape, the content transform, and the frame transform position and clip the facet on the closest drawing environment (the closest canvases in the facet hierarchy which may or may not be the window canvas). During real-time interactions, such as rubber-banding or dragging, your part might need to display directly in the window. In such cases, the window aggregate clip shape, window content transform, and window frame transform specifically position and clip the facet on the window canvas.

The ODFacet class includes several methods that specify geometry (shape and transform objects) or calculate positions on a canvas. Because these calculations necessarily assume a coordinate system, the ODFacet methods include a parameter, *biasCanvas*, that allows you to specify a canvas to whose coordinate space the geometry is biased. The *bias canvas* uses a bias transform to convert from the coordinate system used for drawing on the canvas to the coordinate system (platform-normal coordinates) used by the current graphics system.

On OS/2, facets that are displayed on a window canvas are displayed in separate PM windows. This is of concern to your part only if you want to contain PM controls, such as buttons or scroll bars, or if you want to specify a micro or normal presentation space for your facet's window. See "Facet Windows and Using Embedded Controls" recipe in the *OpenDoc Programming Guide* for more information.

For more information related to bias transform, see the class description for [ODCanvas](#).

## Methods

The methods defined by the ODFacet class include:

- [AcquireActiveShape](#)
- [AcquireAggregateClipShape](#)
- [AcquireClipShape](#)
- [AcquireContentTransform](#)
- [AcquireExternalTransform](#)
- [AcquireFrameAggregateClipShape](#)
- [AcquireFrameTransform](#)
- [AcquireWindowAggregateClipShape](#)
- [AcquireWindowContentTransform](#)
- [AcquireWindowFrameAggregateClipShape](#)
- [AcquireWindowFrameTransform](#)
- [ActiveBorderContainsPoint](#)
- [ChangeActiveShape](#)
- [ChangeCanvas](#)
- [ChangeGeometry](#)
- [ChangeHighlight](#)
- [ContainsPoint](#)
- [CreateCanvas](#)
- [CreateEmbeddedFacet](#)
- [CreateFacetIterator](#)
- [CreatePlatformCanvas](#)
- [CreatePlatformWindowCanvas](#)
- [CreateShape](#)
- [CreateTransform](#)
- [Draw](#)
- [DrawActiveBorder](#)
- [DrawChildren](#)
- [DrawChildrenAlways](#)
- [DrawnIn](#)
- [GetCanvas](#)
- [GetContainingFacet](#)
- [GetFacetHwnd](#)
- [GetFrame](#)
- [GetHandleMouseEvents](#)
- [GetHighlight](#)
- [GetPartInfo](#)
- [GetWindow](#)
- [HasCanvas](#)
- [Invalidate](#)
- [InvalidateActiveBorder](#)
- [IsSelected](#)
- [MoveBefore](#)

- [MoveBehind](#)
- [RemoveFacet](#)
- [SetHandleMouseEvents](#)
- [SetPartInfo](#)
- [SetSelected](#)
- [Update](#)
- [Validate](#)

#### Overridden Methods

There are currently no methods overridden by the ODFacet class.

---

## AcquireActiveShape

---

## AcquireActiveShape - Syntax

This method returns a reference to the active shape for this facet.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODCanvas      *biasCanvas;
ODShape       *rv;

rv = AcquireActiveShape(biasCanvas);
```

---

## AcquireActiveShape Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates.

---

## AcquireActiveShape Return Value - rv

**rv** (ODShape \*) - returns

A reference to the active shape for this facet or the frame shape of this facet's frame if no active shape has been set.

---

## AcquireActiveShape - Parameters

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in platform-normal coordinates.

**rv** (ODShape \*) - returns

A reference to the active shape for this facet or the frame shape of this facet's frame if no active shape has been set.

-----

## AcquireActiveShape - Remarks

This method increments the reference count of the returned shape object. When you have finished using that shape object, you should call its [Release](#) method.

-----

## AcquireActiveShape - Related Methods

### Related Methods

- [ODFacet::ChangeActiveShape](#)

-----

## AcquireActiveShape - Topics

### Class:

ODFacet

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

-----

## AcquireAggregateClipShape

-----

## AcquireAggregateClipShape - Syntax

This method calculates and returns a reference to a shape object that represents the aggregate clip shape of this facet.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODCanvas      *biasCanvas;
ODShape       *rv;

rv = AcquireAggregateClipShape(biasCanvas);
```

---

## AcquireAggregateClipShape Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in platform-normal coordinates.

---

## AcquireAggregateClipShape Return Value - rv

**rv** (ODShape \*) - returns

A reference to a shape object that represents the aggregate clip shape.

---

## AcquireAggregateClipShape - Parameters

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in platform-normal coordinates.

**rv** (ODShape \*) - returns

A reference to a shape object that represents the aggregate clip shape.

---

## AcquireAggregateClipShape - Remarks

The aggregate clip shape describes the facet's clip shape on its canvas. The aggregate clip shape is calculated by intersecting the facet's clip shape with the appropriately transformed clip shapes of all containing facets displayed on this facet's canvas.

This method increments the reference count of the returned shape object. When you have finished using that shape object, you should call its [Release](#) method.

---

## AcquireAggregateClipShape - Related Methods

### Related Methods

- [ODFacet::AcquireClipShape](#)
  - [ODFacet::AcquireWindowAggregateClipShape](#)
- 

## AcquireAggregateClipShape - Topics

**Class:**

ODFacet

Select an item:

[Syntax](#)[Parameters](#)[Returns](#)[Remarks](#)[Related Methods](#)

---

## AcquireClipShape

---

### AcquireClipShape - Syntax

This method returns a reference to this facet's clip shape.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODCanvas      *biasCanvas;
ODShape       *rv;

rv = AcquireClipShape(biasCanvas);
```

---

### AcquireClipShape Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates.

---

### AcquireClipShape Return Value - rv

**rv** (ODShape \*) - returns

A reference to this facet's clip shape.

---

### AcquireClipShape - Parameters

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates.

**rv** (ODShape \*) - returns

A reference to this facet's clip shape.

-----

## AcquireClipShape - Remarks

This method increments the reference count of the returned shape object. When you have finished using that shape object, you should call its [Release](#) method.

-----

## AcquireClipShape - Related Methods

### Related Methods

- [ODFacet::AcquireAggregateClipShape](#)
- [ODFacet::AcquireWindowAggregateClipShape](#)

-----

## AcquireClipShape - Topics

### Class:

ODFacet

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

-----

## AcquireContentTransform

-----

## AcquireContentTransform - Syntax

This method returns a reference to a transform object that represents the content transform of this facet.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>
```

```
ODCanvas      *biasCanvas;
ODTransform   *rv;
```

```
rv = AcquireContentTransform(biasCanvas);
```

---

## AcquireContentTransform Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in platform-normal coordinates.

---

## AcquireContentTransform Return Value - rv

**rv** (ODTransform \*) - returns

A reference to a transform object that represents the content transform.

---

## AcquireContentTransform - Parameters

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in platform-normal coordinates.

**rv** (ODTransform \*) - returns

A reference to a transform object that represents the content transform.

---

## AcquireContentTransform - Remarks

The content transform describes the content coordinate space of this facet on its canvas. The content transform is calculated by concatenating the internal transform of this facet's frame with this facet's external transform and the internal and external transforms of all containing facets displayed on this facet's canvas.

This method increments the reference count of the returned transform object. When you have finished using that transform object, you should call the [Release](#) method.

---

## AcquireContentTransform - Related Methods

### Related Methods

- [ODFacet::AcquireExternalTransform](#)
  - [ODFacet::AcquireFrameTransform](#)
  - [ODFacet::AcquireWindowContentTransform](#)
-

# AcquireContentTransform - Topics

## Class:

ODFacet

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

## AcquireExternalTransform

---

## AcquireExternalTransform - Syntax

This method returns a reference to this facet's external transform.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODCanvas      *biasCanvas;
ODTransform   *rv;

rv = AcquireExternalTransform(biasCanvas);
```

---

## AcquireExternalTransform Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates.

---

## AcquireExternalTransform Return Value - rv

**rv** (ODTransform \*) - returns

A reference to this facet's external transform.

---

## AcquireExternalTransform - Parameters



**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in platform-normal coordinates.

**rv** (ODTransform \*) - returns

A reference to this facet's external transform.

-----

## AcquireExternalTransform - Remarks

This method increments the reference count of the returned transform object. When you have finished using that transform object, you should call its [Release](#) method.

-----

## AcquireExternalTransform - Related Methods

### Related Methods

- [ODFacet::AcquireContentTransform](#)
- [ODFacet::AcquireFrameTransform](#)

-----

## AcquireExternalTransform - Topics

### Class:

ODFacet

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

-----

## AcquireFrameAggregateClipShape (OS/2)

-----

## AcquireFrameAggregateClipShape (OS/2) - Syntax

This method returns a reference to a shape object that represents the facet's aggregate clip shape transformed by the facet's frame transform.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>
```

```
ODCanvas      *biasCanvas;  
ODShape       *rv;  
  
rv = AcquireFrameAggregateClipShape(biasCanvas);
```

---

## AcquireFrameAggregateClipShape (OS/2) Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in platform-normal coordinates.

---

## AcquireFrameAggregateClipShape (OS/2) Return Value - rv

**rv** (ODShape \*) - returns

A reference to a shape object that represents the facet's aggregate clip shape transformed by the facet's frame transform.

---

## AcquireFrameAggregateClipShape (OS/2) - Parameters

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in platform-normal coordinates.

**rv** (ODShape \*) - returns

A reference to a shape object that represents the facet's aggregate clip shape transformed by the facet's frame transform.

---

## AcquireFrameAggregateClipShape (OS/2) - Remarks

This method is provided to allow the facet to cache the shape that is commonly used for setting the clip region for drawing on the canvas.

This method increments the reference count of the returned shape object. When you have finished using that shape object, you should call its [Release](#) method.

---

## AcquireFrameAggregateClipShape (OS/2) - Related Methods

### Related Methods

- [ODFacet::AcquireAggregateClipShape](#)
- [ODFacet::AcquireFrameTransform](#)

---

# AcquireFrameAggregateClipShape (OS/2) - Topics

## Class:

ODFacet

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

## AcquireFrameTransform

---

## AcquireFrameTransform - Syntax

This method returns a reference to a transform object that represents the frame transform of this facet.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODCanvas      *biasCanvas;
ODTransform   *rv;

rv = AcquireFrameTransform(biasCanvas);
```

---

## AcquireFrameTransform Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates.

---

## AcquireFrameTransform Return Value - rv

**rv** (ODTransform \*) - returns

A reference to a transform object that represents the frame transform.

---

## AcquireFrameTransform - Parameters

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in platform-normal coordinates.

**rv** (ODTransform \*) - returns

A reference to a transform object that represents the frame transform.

-----

## AcquireFrameTransform - Remarks

The frame transform describes the frame coordinate space of this facet on its canvas. The frame transform is calculated by concatenating this facet's external transform with the internal and external transforms of all containing facets displayed on this facet's canvas.

This method increments the reference count of the returned transform object. When you have finished using that transform object, you should call its [Release](#) method.

-----

## AcquireFrameTransform - Related Methods

### Related Methods

- [ODFacet::AcquireContentTransform](#)
- [ODFacet::AcquireExternalTransform](#)
- [ODFacet::AcquireWindowFrameTransform](#)

-----

## AcquireFrameTransform - Topics

### Class:

ODFacet

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

-----

## AcquireWindowAggregateClipShape

-----

## AcquireWindowAggregateClipShape - Syntax

This method returns a reference to a shape object that represents the window aggregate clip shape of this facet.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODCanvas      *biasCanvas;
ODShape       *rv;

rv = AcquireWindowAggregateClipShape(biasCanvas);
```

---

## AcquireWindowAggregateClipShape Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in the platform-normal coordinates.

---

## AcquireWindowAggregateClipShape Return Value - rv

**rv** (ODShape \*) - returns

A reference to a shape object that represents the window aggregate clip shape.

---

## AcquireWindowAggregateClipShape - Parameters

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in the platform-normal coordinates.

**rv** (ODShape \*) - returns

A reference to a shape object that represents the window aggregate clip shape.

---

## AcquireWindowAggregateClipShape - Remarks

The window aggregate clip shape describes the facet's clip shape on its window. The window aggregate clip shape is calculated by intersecting the clip shape of this facet with the clip shapes of all containing facets displayed on this facet's window canvas.

This method increments the reference count of the returned shape object. When you have finished using that shape object, you should call the [Release](#) method.

---

## AcquireWindowAggregateClipShape - Related Methods

### Related Methods

- [ODFacet::AcquireAggregateClipShape](#)

---

## AcquireWindowAggregateClipShape - Topics

### Class:

ODFacet

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

## AcquireWindowContentTransform

---

## AcquireWindowContentTransform - Syntax

This method returns a reference to a transform object that represents the window-content transform of this facet.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODCanvas      *biasCanvas;
ODTransform   *rv;

rv = AcquireWindowContentTransform(biasCanvas);
```

---

## AcquireWindowContentTransform Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates.

---

## AcquireWindowContentTransform Return Value - rv

**rv** (ODTransform \*) - returns

A reference to a transform object that represents the window-content transform.

---

## AcquireWindowContentTransform - Parameters

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in platform-normal coordinates.

**rv** (ODTransform \*) - returns

A reference to a transform object that represents the window-content transform.

---

## AcquireWindowContentTransform - Remarks

The window-content transform describes the content coordinate space of this facet on its window. The window-content transform is calculated by concatenating the internal transform of this facet's frame with this facet's external transform and the internal and external transforms of all containing facets displayed on this facet's window canvas.

This method increments the reference count of the returned transform object. When you have finished using that transform object, you should call its [Release](#) method.

---

## AcquireWindowContentTransform - Related Methods

### Related Methods

- [ODFacet::AcquireContentTransform](#)
  - [ODFacet::AcquireWindowFrameTransform](#)
- 

## AcquireWindowContentTransform - Topics

### Class:

ODFacet

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## AcquireWindowFrameAggregateClipShape (OS/2)

---

## AcquireWindowFrameAggregateClipShape (OS/2) - Syntax

This method returns a reference to a shape object that represents the facet's aggregate clip shape transformed by the facet's window frame transform.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODCanvas      *biasCanvas;
ODShape       *rv;

rv = AcquireWindowFrameAggregateClipShape(
    biasCanvas);
```

-----

## AcquireWindowFrameAggregateClipShape (OS/2) Parameter - bias

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates..

-----

## AcquireWindowFrameAggregateClipShape (OS/2) Return Value - rv

**rv** (ODShape \*) - returns

A reference to a shape object that represents the facet's aggregate clip shape transformed by the facet's window frame transform.

-----

## AcquireWindowFrameAggregateClipShape (OS/2) - Parameters

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates..

**rv** (ODShape \*) - returns

A reference to a shape object that represents the facet's aggregate clip shape transformed by the facet's window frame transform.

-----

## AcquireWindowFrameAggregateClipShape (OS/2) - Remarks

This method allows the facet to cache the shape that is commonly used for setting up the clip region for drawing on the window canvas.

This method increments the reference count of the returned shape object. When you have finished using that shape object, you should call its [Release](#) method.

-----

## AcquireWindowFrameAggregateClipShape (OS/2) - Topics



**Class:**  
ODFacet

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

## AcquireWindowFrameTransform

---

### AcquireWindowFrameTransform - Syntax

This method returns a reference to a transform object that represents the window-frame transform of this facet.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODCanvas      *biasCanvas;
ODTransform    *rv;

rv = AcquireWindowFrameTransform(biasCanvas);
```

---

### AcquireWindowFrameTransform Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input  
A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates.

---

### AcquireWindowFrameTransform Return Value - rv

**rv** (ODTransform \*) - returns  
A reference to a transform object that represents the window-frame transform.

---

### AcquireWindowFrameTransform - Parameters

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in platform-normal coordinates.

**rv** (ODTransform \*) - returns

A reference to a transform object that represents the window-frame transform.

-----

## AcquireWindowFrameTransform - Remarks

The window-frame transform describes the frame coordinate space of this facet on its window. The window-frame transform is calculated by concatenating this facet's external transform with the internal and external transforms of all containing facets displayed on this facet's window canvas.

This method increments the reference count of the returned transform object. When you have finished using that transform object, you should call its [Release](#) method.

-----

## AcquireWindowFrameTransform - Related Methods

### Related Methods

- [ODFacet::AcquireFrameTransform](#)
- [ODFacet::AcquireWindowContentTransform](#)

-----

## AcquireWindowFrameTransform - Topics

### Class:

ODFacet

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

-----

## ActiveBorderContainsPoint

-----

## ActiveBorderContainsPoint - Syntax

This method indicates whether this facet's frame is active and whether a point is within its active frame border.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>
```

```
ODPoint      *point;
ODCanvas     *biasCanvas;
ODBoolean    rv;

rv = ActiveBorderContainsPoint(point, biasCanvas);
```

-----

## ActiveBorderContainsPoint Parameter - point

**point** (ODPoint \*) - input  
The location to be tested, expressed in frame coordinates.

-----

## ActiveBorderContainsPoint Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input  
A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in platform-normal coordinates.

-----

## ActiveBorderContainsPoint Return Value - rv

**rv** (ODBoolean) - returns  
A flag indicating whether this facet's frame is active and whether the specified point is in the active frame border.

kODTrue	The facet's frame is active and the specified point is within its active frame border.
kODFalse	The facet's frame is not active or the specified point is not within its active frame border.

-----

## ActiveBorderContainsPoint - Parameters

**point** (ODPoint \*) - input  
The location to be tested, expressed in frame coordinates.

**biasCanvas** (ODCanvas \*) - input  
A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in platform-normal coordinates.

**rv** (ODBoolean) - returns  
A flag indicating whether this facet's frame is active and whether the specified point is in the active frame border.

kODTrue	The facet's frame is active and the specified point is within its active frame border.
kODFalse	The facet's frame is not active or the specified point is not within its active frame border.

-----

# ActiveBorderContainsPoint - Remarks

Your part calls its embedded facet's `ActiveBorderContainsPoint` method when it receives a `WM_MOUSEMOVE` event to determine if it should change the pointer shape. You should not change the pointer shape if the mouse is over the active border of an embedded frame.

OpenDoc draws this facet's active frame border on the border of the active shape associated with this facet. The active frame border is derived from the active shape, but is not identical to it.

---

# ActiveBorderContainsPoint - Related Methods

## Related Methods

- [ODFacet::ContainsPoint](#)
- 

# ActiveBorderContainsPoint - Topics

## Class:

ODFacet

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

# ChangeActiveShape

---

# ChangeActiveShape - Syntax

This method assigns the specified shapes as the active shape of this facet.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>
```

```
ODShape      *activeShape;
ODCanvas     *biasCanvas;
```

```
ChangeActiveShape(activeShape, biasCanvas);
```

---

# ChangeActiveShape Parameter - activeShape

**activeShape** (ODShape \*) - input

A reference to a shape to be assigned as this facet's active shape.

---

## ChangeActiveShape Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in platform-normal coordinates.

---

## ChangeActiveShape - Return Value

None.

---

## ChangeActiveShape - Parameters

**activeShape** (ODShape \*) - input

A reference to a shape to be assigned as this facet's active shape.

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in platform-normal coordinates.

None.

---

## ChangeActiveShape - Remarks

When this facet's frame has the selection focus, OpenDoc automatically takes care of changing the facet's active border shape and redrawing the active frame border; otherwise, only this facet's part should change this facet's active shape.

---

## ChangeActiveShape - Related Methods

### Related Methods

- [ODFacet::AcquireActiveShape](#)
-

# ChangeActiveShape - Topics

## Class:

ODFacet

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

## ChangeCanvas

---

## ChangeCanvas - Syntax

This method attaches a canvas to this facet.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>
```

```
ODCanvas      *canvas;
```

```
ChangeCanvas (canvas) ;
```

---

## ChangeCanvas Parameter - canvas

**canvas** (ODCanvas \*) - input

A reference to a canvas to be attached to this facet.

---

## ChangeCanvas - Return Value

None.

---

## ChangeCanvas - Parameters

**canvas** (ODCanvas \*) - input

A reference to a canvas to be attached to this facet.

None.

---

## ChangeCanvas - Remarks

This method in turn calls its canvas's [SetFacet](#) method to ensure the canvas has a reference back to this facet. After this method executes successfully, you can display this facet and its embedded facets on the specified canvas.

It is the caller's responsibility to deallocate the canvas's storage when it is no longer needed.

---

## ChangeCanvas - Related Methods

### Related Methods

- [ODCanvas::SetFacet](#)
  - [ODFacet::GetCanvas](#)
  - [ODFacet::HasCanvas](#)
- 

## ChangeCanvas - Topics

### Class:

ODFacet

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## ChangeGeometry

---

## ChangeGeometry - Syntax

This method assigns the specified clip shape, external transform, or both to this facet.

```
#define INCL_ODFACET
```

```
#define INCL_ODAPI
#include <os2.h>

ODShape      *clipShape;
ODTransform  *transform;
ODCanvas     *biasCanvas;

ChangeGeometry(clipShape, transform, biasCanvas);
```

---

## ChangeGeometry Parameter - clipShape

**clipShape** (ODShape \*) - input  
A reference to a shape to be assigned as this facet's clip shape.

---

## ChangeGeometry Parameter - transform

**transform** (ODTransform \*) - input  
A reference to a transform to be assigned as this facet's external transform.

---

## ChangeGeometry Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input  
A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates.

---

## ChangeGeometry - Return Value

None.

---

## ChangeGeometry - Parameters

**clipShape** (ODShape \*) - input  
A reference to a shape to be assigned as this facet's clip shape.

**transform** (ODTransform \*) - input  
A reference to a transform to be assigned as this facet's external transform.

**biasCanvas** (ODCanvas \*) - input  
A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal



coordinates.

None.

-----

## ChangeGeometry - Remarks

This method invalidates any cached aggregate clip shape and transforms. This facet's containing part calls this method to change the facet's clip shape, external transform, or both. This method in turn calls the [GeometryChanged](#) method associated with this facet's part to notify the part (and parts of all its embedded facets) that its clip shape, external transform, or both has changed.

-----

## ChangeGeometry - Related Methods

### Related Methods

- [ODPart::GeometryChanged](#)

-----

## ChangeGeometry - Topics

### Class:

ODFacet

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

-----

## ChangeHighlight

-----

## ChangeHighlight - Syntax

This method changes the highlight state of this facet.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODHighlight highlight;

ChangeHighlight (highlight);
```

---

## ChangeHighlight Parameter - highlight

**highlight** ([ODHighlight](#)) - input

The highlight state to be assigned to this facet. This parameter can be set to one of the following values:

kODNoHighlight	The facet is not highlighted.
kODFullHighlight	The facet is highlighted in foreground style.
kODDimHighlight	The facet is highlighted in background style.

---

## ChangeHighlight - Return Value

None.

---

## ChangeHighlight - Parameters

**highlight** ([ODHighlight](#)) - input

The highlight state to be assigned to this facet. This parameter can be set to one of the following values:

kODNoHighlight	The facet is not highlighted.
kODFullHighlight	The facet is highlighted in foreground style.
kODDimHighlight	The facet is highlighted in background style.

None.

---

## ChangeHighlight - Remarks

The facet's containing part calls this method to change the facet's highlight state (usually so that its embedded part's highlighting corresponds to that of its containing part). This method in turn calls the [HighlightChanged](#) method associated with this facet's part to notify the part that its highlight state has changed.

---

## ChangeHighlight - Related Methods

### Related Methods

- [ODFacet::GetHighlight](#)
- [ODPart::HighlightChanged](#)

---

## ChangeHighlight - Topics

### Class:

ODFacet

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

## ContainsPoint

---

## ContainsPoint - Syntax

This method indicates whether the specified point is within the area of this facet.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODPoint      *point;
ODCanvas     *biasCanvas;
ODBoolean     rv;

rv = ContainsPoint(point, biasCanvas);
```

---

## ContainsPoint Parameter - point

**point** ([ODPoint \\*](#)) - input

The location to be tested, expressed in frame coordinates.

---

## ContainsPoint Parameter - biasCanvas

**biasCanvas** ([ODCanvas \\*](#)) - input

A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates.

---

## ContainsPoint Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the specified point is within the area of this facet.

kODTrue

The specified point is within the area of this facet.

kODFalse

The specified point is outside the area of this facet.

---

## ContainsPoint - Parameters

**point** ([ODPoint \\*](#)) - input

The location to be tested, expressed in frame coordinates.

**biasCanvas** ([ODCanvas \\*](#)) - input

A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in platform-normal coordinates.

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the specified point is within the area of this facet.

kODTrue

The specified point is within the area of this facet.

kODFalse

The specified point is outside the area of this facet.

---

## ContainsPoint - Remarks

Your part calls its embedded facet's ContainsPoint method when it needs to do hit-testing on those facets. This method tests the result against the intersection of the clip shape and the active shape.

---

## ContainsPoint - Related Methods

### Related Methods

- [ODFacet::ActiveBorderContainsPoint](#)
- 

## ContainsPoint - Topics

### Class:

ODFacet

Select an item:

---

# CreateCanvas

---

## CreateCanvas - Syntax

This method creates a canvas object.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODGraphicsSystem    graphicsSystem;
ODPlatformCanvas    *platformCanvas;
ODBoolean            isDynamic;
ODBoolean            isOffscreen;
ODCanvas             *rv;

rv = CreateCanvas(graphicsSystem, platformCanvas,
                  isDynamic, isOffscreen);
```

---

## CreateCanvas Parameter - graphicsSystem

**graphicsSystem** ([ODGraphicsSystem](#)) - input

The graphics system you want to use for the canvas. Valid graphics systems are platform-dependent. On OS/2, this parameter should always be set to KODGPI.

---

## CreateCanvas Parameter - platformCanvas

**platformCanvas** ([ODPlatformCanvas \\*](#)) - input

A reference to the graphics-system-specific drawing canvas to be assigned to this canvas or KODNULL if there is no drawing structure. Valid values for this parameter are graphics-system dependent.

---

## CreateCanvas Parameter - isDynamic

**isDynamic** ([ODBoolean](#)) - input

A flag indicating whether this canvas is to be dynamic.

kODTrue

This canvas is to be dynamic.

kODFalse

This canvas is to be static.

-----

## CreateCanvas Parameter - isOffscreen

**isOffscreen** (ODBoolean) - input

A flag indicating whether this canvas is to be offscreen.

kODTrue

This canvas is to be offscreen.

kODFalse

This canvas is to be onscreen.

-----

## CreateCanvas Return Value - rv

**rv** (ODCanvas \*) - returns

A reference to a new canvas object.

-----

## CreateCanvas - Parameters

**graphicsSystem** (ODGraphicsSystem) - input

The graphics system you want to use for the canvas. Valid graphics systems are platform-dependent. On OS/2, this parameter should always be set to kODGPI.

**platformCanvas** (ODPlatformCanvas \*) - input

A reference to the graphics-system-specific drawing canvas to be assigned to this canvas or kODNULL if there is no drawing structure. Valid values for this parameter are graphics-system dependent.

**isDynamic** (ODBoolean) - input

A flag indicating whether this canvas is to be dynamic.

kODTrue

This canvas is to be dynamic.

kODFalse

This canvas is to be static.

**isOffscreen** (ODBoolean) - input

A flag indicating whether this canvas is to be offscreen.

kODTrue

This canvas is to be offscreen.

kODFalse

This canvas is to be onscreen.

**rv** (ODCanvas \*) - returns

A reference to a new canvas object.

-----

## CreateCanvas - Remarks

Your part calls this method to create an offscreen canvas object to attach to this facet. To create a canvas that will not be attached to any facet, call the window-state object's [CreateCanvas](#) method.

---

## CreateCanvas - Related Methods

### Related Methods

- [ODWindowState::CreateCanvas](#)
- 

## CreateCanvas - Topics

### Class:

ODFacet

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## CreateEmbeddedFacet

---

## CreateEmbeddedFacet - Syntax

This method creates a facet for the specified frame, embedded in this facet.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODFrame          *frame;
ODShape          *clipShape;
ODTransform      *externalTransform;
ODCanvas         *canvas;
ODCanvas         *biasCanvas;
ODFacet          *siblingFacet;
ODFramePosition  position;
ODFacet          *rv;

rv = CreateEmbeddedFacet(frame, clipShape,
                        externalTransform, canvas, biasCanvas,
                        siblingFacet, position);
```

---

## CreateEmbeddedFacet Parameter - frame

**frame** (ODFrame \*) - input  
A reference to a frame for this facet.

---

## CreateEmbeddedFacet Parameter - clipShape

**clipShape** (ODShape \*) - input  
A reference to an initial clip shape for this facet.

---

## CreateEmbeddedFacet Parameter - externalTransform

**externalTransform** (ODTransform \*) - input  
A reference to an initial external transform for this facet.

---

## CreateEmbeddedFacet Parameter - canvas

**canvas** (ODCanvas \*) - input  
A reference to the canvas this facet should draw to or KODNULL if identical to the canvas associated with the containing facet.

---

## CreateEmbeddedFacet Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input  
A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates.

---

## CreateEmbeddedFacet Parameter - siblingFacet

**siblingFacet** (ODFacet \*) - input  
A reference to an existing embedded facet of this facet or KODNULL if the embedded facet does not exist.

---

## CreateEmbeddedFacet Parameter - position



**position** ([ODFramePosition](#)) - input

The desired position of the new facet relative to the sibling facet. This parameter can be set to one of the following values:

kODFrameBehind

This frame is behind its sibling.

kODFrameInFront

This frame is in front of its sibling.

kODNULL

The new facet is placed in front of or behind all its sibling facets.

-----

## CreateEmbeddedFacet Return Value - rv

**rv** (ODFacet \*) - returns

A reference to a new facet object.

-----

## CreateEmbeddedFacet - Parameters

**frame** (ODFrame \*) - input

A reference to a frame for this facet.

**clipShape** (ODShape \*) - input

A reference to an initial clip shape for this facet.

**externalTransform** (ODTransform \*) - input

A reference to an initial external transform for this facet.

**canvas** (ODCanvas \*) - input

A reference to the canvas this facet should draw to or kODNULL if identical to the canvas associated with the containing facet.

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in platform-normal coordinates.

**siblingFacet** (ODFacet \*) - input

A reference to an existing embedded facet of this facet or kODNULL if the embedded facet does not exist.

**position** ([ODFramePosition](#)) - input

The desired position of the new facet relative to the sibling facet. This parameter can be set to one of the following values:

kODFrameBehind

This frame is behind its sibling.

kODFrameInFront

This frame is in front of its sibling.

kODNULL

The new facet is placed in front of or behind all its sibling facets.

**rv** (ODFacet \*) - returns

A reference to a new facet object.

-----

## CreateEmbeddedFacet - Remarks

The facet's part calls this method to create a facet for one of its embedded frames. This method in turn calls its frame's [FacetAdded](#) method, which in turn calls the [FacetAdded](#) method associated with the frame's part to notify the part that a facet has been added to one of its display frames.

-----

# CreateEmbeddedFacet - Exception Handling

kODErrInvalidFacet

The specified sibling facet is not an embedded facet of this facet.

kODErrUnsupportedFramePositionCode

The specified designated position is not a valid position code.

---

## CreateEmbeddedFacet - Related Methods

### Related Methods

- [ODFrame::FacetAdded](#)
- [ODPart::FacetAdded](#)

---

## CreateEmbeddedFacet - Topics

### Class:

[ODFacet](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## CreateFacetIterator

---

## CreateFacetIterator - Syntax

This method creates a facet iterator object for the embedded facets of this facet.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODTraversalType traversalType;
ODSiblingOrder siblingOrder;
ODFacetIterator *rv;

rv = CreateFacetIterator(traversalType, siblingOrder);
```

---

## CreateFacetIterator Parameter - traversalType

**traversalType** ([ODTraversalType](#)) - input

The traversal type to be assigned to the facet iterator. This parameter can be set to one of the following values:

- kODBottomUp**      The facet hierarchy is traversed bottom up, visiting this facet after visiting all its children.
- kODChildrenOnly**      Only the children of the specified facet are traversed (not including this facet itself).
- kODTopDown**      The facet hierarchy is traversed top down, visiting this facet and then all of its children.

---

## CreateFacetIterator Parameter - siblingOrder

**siblingOrder** ([ODSiblingOrder](#)) - input

The order in which the iterator traverses the sibling facets. This parameter can be set to one of the following values:

- kODBackToFront**      The siblings in a facet hierarchy are processed from back to front.
- kODFrontToBack**      The siblings in a facet hierarchy are processed from front to back.

---

## CreateFacetIterator Return Value - rv

**rv** ([ODFacetIterator \\*](#)) - returns

A reference to a new facet iterator object.

---

## CreateFacetIterator - Parameters

**traversalType** ([ODTraversalType](#)) - input

The traversal type to be assigned to the facet iterator. This parameter can be set to one of the following values:

- kODBottomUp**      The facet hierarchy is traversed bottom up, visiting this facet after visiting all its children.
- kODChildrenOnly**      Only the children of the specified facet are traversed (not including this facet itself).
- kODTopDown**      The facet hierarchy is traversed top down, visiting this facet and then all of its children.

**siblingOrder** ([ODSiblingOrder](#)) - input

The order in which the iterator traverses the sibling facets. This parameter can be set to one of the following values:

kODBackToFront

The siblings in a facet hierarchy are processed from back to front.

kODFrontToBack

The siblings in a facet hierarchy are processed from front to back.

**rv** (ODFacetIterator \*) - returns

A reference to a new facet iterator object.

---

## CreateFacetIterator - Remarks

Your part calls this method if it needs to apply an operation to a facet and all its embedded facets. For example, OpenDoc might use a facet iterator to make sure that all facets within a facet being invalidated are also invalidated. It is your responsibility to delete the iterator when it is no longer needed.

While you are using a facet iterator, you should not modify the list of embedded facets. You must postpone adding items to or removing items from the list of embedded facets until after you have deleted the iterator.

---

## CreateFacetIterator - Related Methods

### Related Methods

- [ODFrame::CreateFacetIterator](#)

---

## CreateFacetIterator - Topics

### Class:

ODFacet

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

## CreatePlatformCanvas (OS/2)

---

## CreatePlatformCanvas (OS/2) - Syntax

This method creates and initializes a platform canvas for an offscreen static, offscreen dynamic, or onscreen static canvas, using the specified presentation space handle.

```
#define INCL_ODFACET
```

```
#define INCL_ODAPI
#include <os2.h>

HPS          hps;
ODPlatformCanvas *rv;

rv = CreatePlatformCanvas(hps);
```

-----

## CreatePlatformCanvas (OS/2) Parameter - hps

**hps** ([HPS](#)) - input  
A handle to the presentation space created using the GpiCreatePS function.

-----

## CreatePlatformCanvas (OS/2) Return Value - rv

**rv** ([ODPlatformCanvas \\*](#)) - returns  
A newly created and initialized platform canvas object. This object can be specified as an input parameter to the facet's [CreateCanvas](#) method.

-----

## CreatePlatformCanvas (OS/2) - Parameters

**hps** ([HPS](#)) - input  
A handle to the presentation space created using the GpiCreatePS function.

**rv** ([ODPlatformCanvas \\*](#)) - returns  
A newly created and initialized platform canvas object. This object can be specified as an input parameter to the facet's [CreateCanvas](#) method.

-----

## CreatePlatformCanvas (OS/2) - Related Methods

### Related Methods

- [ODFacet::CreateCanvas](#)

-----

## CreatePlatformCanvas (OS/2) - Topics

**Class:**  
ODFacet

Select an item:  
[Syntax](#)

---

## CreatePlatformWindowCanvas (OS/2)

---

### CreatePlatformWindowCanvas (OS/2) - Syntax

This method creates and initializes an platform canvas for an onscreen dynamic canvas using the specified window object.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODWindow          *window;
ODPlatformCanvas  *rv;

rv = CreatePlatformWindowCanvas(window);
```

---

### CreatePlatformWindowCanvas (OS/2) Parameter - window

**window** (ODWindow \*) - input  
The window object to be used with this canvas.

---

### CreatePlatformWindowCanvas (OS/2) Return Value - rv

**rv** (ODPlatformCanvas \*) - returns  
A newly created and initialized platform canvas object. This object can be specified as an input parameter to the facet's [CreateCanvas](#) method.

---

### CreatePlatformWindowCanvas (OS/2) - Parameters

**window** (ODWindow \*) - input  
The window object to be used with this canvas.

**rv** (ODPlatformCanvas \*) - returns  
A newly created and initialized platform canvas object. This object can be specified as an input parameter to the facet's [CreateCanvas](#) method.

---

## CreatePlatformWindowCanvas (OS/2) - Remarks

The object returned by this method is an instance of the [ODPlatformWindowCanvas](#) class; however, because [ODPlatformWindowCanvas](#) only modifies behavior and does not introduce any new methods, the object is referred to using its base class ([ODPlatformCanvas](#)).

---

## CreatePlatformWindowCanvas (OS/2) - Topics

### Class:

ODFacet

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

---

## CreateShape

---

## CreateShape - Syntax

This method creates a shape object.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>
```

```
ODShape      *rv;
```

```
rv = CreateShape();
```

---

## CreateShape Return Value - rv

**rv** (ODShape \*) - returns

A reference to a new shape object.

---

## CreateShape - Parameters

**rv** (ODShape \*) - returns  
A reference to a new shape object.

---

## CreateShape - Remarks

Your part calls this method to create a shape object for any purpose.

This method initializes the reference count of the returned shape object. When you have finished using that shape object, you should call its [Release](#) method.

---

## CreateShape - Related Methods

### Related Methods

- [ODFrame::CreateShape](#)
- 

## CreateShape - Topics

### Class:

ODFacet

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## CreateTransform

---

## CreateTransform - Syntax

This method creates a transform object.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODTransform      *rv;

rv = CreateTransform();
```



---

## CreateTransform Return Value - rv

**rv** (ODTransform \*) - returns  
A reference to a new transform object.

---

## CreateTransform - Parameters

**rv** (ODTransform \*) - returns  
A reference to a new transform object.

---

## CreateTransform - Remarks

Your part calls this method to create a transform object for any purpose.

This method initializes the reference count of the returned transform object. When you have finished using that transform object, you should call its [Release](#) method.

---

## CreateTransform - Related Methods

### Related Methods

- [ODFrame::CreateTransform](#)
- 

## CreateTransform - Topics

**Class:**  
ODFacet

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## Draw

---

## Draw - Syntax

This method tells this facet's part to draw itself within the specified portion in this facet.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODShape      *invalidShape;
ODCanvas     *biasCanvas;

Draw(invalidShape, biasCanvas);
```

---

## Draw Parameter - invalidShape

**invalidShape** (ODShape \*) - input

A reference to a shape within which the part should draw itself, expressed in frame coordinates.

---

## Draw Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates.

---

## Draw - Return Value

None.

---

## Draw - Parameters

**invalidShape** (ODShape \*) - input

A reference to a shape within which the part should draw itself, expressed in frame coordinates.

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates.

None.

---

## Draw - Remarks

This facet's containing part calls this method when the embedded facets of this facet require updating. This method in turn calls the [Draw](#) method associated with this facet's part.

---

## Draw - Related Methods

### Related Methods

- [ODFacet::DrawChildren](#)
  - [ODFacet::DrawChildrenAlways](#)
  - [ODPart::Draw](#)
- 

## Draw - Topics

### Class:

ODFacet

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## DrawActiveBorder

---

## DrawActiveBorder - Syntax

This method updates the active frame border for this facet.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>
```

```
DrawActiveBorder();
```

---

## DrawActiveBorder - Return Value

None.

---

## DrawActiveBorder - Parameters

None.

---

## DrawActiveBorder - Remarks

OpenDoc calls this method. When this facet's frame has the selection focus, OpenDoc automatically takes care of changing the facet's active border shape and redrawing the active frame border.

Under normal circumstances, there is no need to invalidate the active frame border for either the active part or its container part; however, parts could call this method if they want to force the active frame border to be redrawn. To do so, the part must invalidate the active frame border by calling its facet's [InvalidateActiveBorder](#) method and redraw the active frame border by calling its facet's [DrawActiveBorder](#) method.

---

## DrawActiveBorder - Related Methods

### Related Methods

- [ODFacet::InvalidateActiveBorder](#)
  - [ODFrame::DrawActiveBorder](#)
- 

## DrawActiveBorder - Topics

### Class:

ODFacet

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## DrawChildren

---

# DrawChildren - Syntax

This method draws all embedded facets of this facet that need updating.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODShape      *invalidShape;
ODCanvas     *biasCanvas;

DrawChildren(invalidShape, biasCanvas);
```

---

## DrawChildren Parameter - invalidShape

**invalidShape** (ODShape \*) - input

A reference to a shape within which the embedded facets should draw, expressed in frame coordinates.

---

## DrawChildren Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates.

---

## DrawChildren - Return Value

None.

---

## DrawChildren - Parameters

**invalidShape** (ODShape \*) - input

A reference to a shape within which the embedded facets should draw, expressed in frame coordinates.

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates.

None.

---

## DrawChildren - Remarks

This facet's containing part calls this method when the embedded facets of this facet require updating. This method in turn calls the [Draw](#) method associated with this facet's embedded part.

---

## DrawChildren - Related Methods

### Related Methods

- [ODFacet::DrawChildrenAlways](#)
  - [ODPart::Draw](#)
- 

## DrawChildren - Topics

### Class:

ODFacet

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## DrawChildrenAlways

---

## DrawChildrenAlways - Syntax

This method draws all embedded facets of this facet, whether they need updating or not.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODShape      *invalidShape;
ODCanvas     *biasCanvas;

DrawChildrenAlways(invalidShape, biasCanvas);
```

---

## DrawChildrenAlways Parameter - invalidShape

**invalidShape** (ODShape \*) - input

A reference to a shape within which the embedded facets should draw, expressed in frame coordinates.

---

## DrawChildrenAlways Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates.

---

## DrawChildrenAlways - Return Value

None.

---

## DrawChildrenAlways - Parameters

**invalidShape** (ODShape \*) - input

A reference to a shape within which the embedded facets should draw, expressed in frame coordinates.

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates.

None.

---

## DrawChildrenAlways - Remarks

This facet's containing part calls this method to draw all embedded facets of this facet, whether they need updating or not. This method in turn calls the [Draw](#) method associated with this facet's embedded part.

---

## DrawChildrenAlways - Related Methods

### Related Methods

- [ODFacet::DrawChildren](#)
- [ODPart::Draw](#)

---

# DrawChildrenAlways - Topics

## Class:

ODFacet

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

## DrawnIn

---

## DrawnIn - Syntax

This method is called when this facet has been drawn in without an update event being generated.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODShape      *shape;
ODCanvas     *biasCanvas;

DrawnIn(shape, biasCanvas);
```

---

## DrawnIn Parameter - shape

**shape** (ODShape \*) - input

A reference to a shape object defining the area in this facet that was updated, expressed in frame coordinates.

---

## DrawnIn Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates.

---

## DrawnIn - Return Value



None.

-----

## DrawnIn - Parameters

**shape** (ODShape \*) - input

A reference to a shape object defining the area in this facet that was updated, expressed in frame coordinates.

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in platform-normal coordinates.

None.

-----

## DrawnIn - Remarks

This facet's part calls this method when it has drawn asynchronously to the facet. This method in turn calls the [CanvasUpdated](#) method associated with the owner of this facet's canvas to notify it that its canvas has changed. The owning part decides when to process the update.

-----

## DrawnIn - Exception Handling

kODErrFacetNotFound

The requested facet was not found.

-----

## DrawnIn - Related Methods

### Related Methods

- [ODPart::CanvasUpdated](#)

-----

## DrawnIn - Topics

### Class:

ODFacet

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

## GetCanvas

---

## GetCanvas - Syntax

This method returns a reference to the canvas associated with this facet.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODCanvas      *rv;

rv = GetCanvas();
```

---

## GetCanvas Return Value - rv

**rv** (ODCanvas \*) - returns  
A reference to the canvas associated with this facet.

---

## GetCanvas - Parameters

**rv** (ODCanvas \*) - returns  
A reference to the canvas associated with this facet.

---

## GetCanvas - Remarks

If this facet has no canvas of its own, this method searches recursively for the containing facet's canvas.

---

## GetCanvas - Exception Handling

kODErrInvalidCanvas

Neither this facet nor any of its containing facets has a canvas.

---

## GetCanvas - Related Methods

### Related Methods

- [ODFacet::ChangeCanvas](#)
  - [ODFacet::HasCanvas](#)
- 

## GetCanvas - Topics

### Class:

[ODFacet](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

## GetContainingFacet

---

## GetContainingFacet - Syntax

This method returns a reference to the containing facet of this facet.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODFacet      *rv;

rv = GetContainingFacet();
```

---

## GetContainingFacet Return Value - rv

**rv** (ODFacet \*) - returns

A reference to the containing facet of this facet or kODNULL if the facet has no containing facet.

---

## GetContainingFacet - Parameters

**rv** (ODFacet \*) - returns  
A reference to the containing facet of this facet or kODNULL if the facet has no containing facet.

---

## GetContainingFacet - Topics

**Class:**  
ODFacet

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## GetFacetHWND (OS/2)

---

## GetFacetHWND (OS/2) - Syntax

This method returns a handle to the Presentation Manager (PM) window associated with this facet.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>
```

```
HWND    rv;
```

```
rv = GetFacetHWND();
```

---

## GetFacetHWND (OS/2) Return Value - rv

**rv** ([HWND](#)) - returns  
A handle to the PM window associated with this facet or 0 if the facet does not have a window.

---

## GetFacetHWND (OS/2) - Parameters

**rv** ([HWND](#)) - returns

A handle to the PM window associated with this facet or 0 if the facet does not have a window.

---

## GetFacetHWND (OS/2) - Remarks

Facet windows are created and managed by OpenDoc for facets that descend from a root facet with a window (onscreen) canvas. In general, parts should not attempt to manipulate the facet window; however, the [HWND](#) for the facet window is provided so that it can be used as the parent and owner of embedded PM controls, such as buttons and scroll bars. Notification messages sent to the facet window are forwarded to the facet's part in that part's [HandleEvent](#) method. Parts can determine that an event came from the facet window rather than the OpenDoc dispatcher by examining the *hwnd* field of the [ODEventData](#) data structure. If the event came from the facet window, this field contains the window handle of the facet.

Just because a facet has a facet window does not mean that drawing should be directed to the window. The facet's containing facet, or any of its ancestors, can have an offscreen canvas set. In this case, the facet window is not shown and any drawing directed to the window is not seen. Drawing should generally be directed to the presentation space obtained from the facet's platform canvas object.

---

## GetFacetHWND (OS/2) - Topics

### Class:

ODFacet

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

---

## GetFrame

---

## GetFrame - Syntax

This method returns a reference to this facet's frame.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>
```

```
ODFrame      *rv;
```

```
rv = GetFrame();
```

---

## GetFrame Return Value - rv

**rv** (ODFrame \*) - returns  
A reference to this facet's frame.

---

## GetFrame - Parameters

**rv** (ODFrame \*) - returns  
A reference to this facet's frame.

---

## GetFrame - Remarks

This method increments the reference count of the returned frame object. When you have finished using that frame object, you should call its [Release](#) method.

---

## GetFrame - Topics

**Class:**  
ODFacet

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

## GetHandleMouseEvents (OS/2)

---

## GetHandleMouseEvents (OS/2) - Syntax

This method returns the flags used to determine how mouse events are dispatched to PM controls, such as buttons and scroll bars.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>
```

```
ODULong    rv;
```

```
rv = GetHandleMouseEvents();
```

---

# GetHandleMouseEvents (OS/2) Return Value - rv

**rv** ([ODULong](#)) - returns

A bit field which can contain one or both of the following flags ORed together.

kODPartHandlesDragDropEvents

Drag-and-drop events which occur over PM windows which are children of this facet's facet window are handled by this facet's part in its [DragEnter](#), [DragWithin](#), and [DragLeave](#) methods.

kODPartHandlesMouseEvents

Mouse events (excluding drag-and-drop events) which occur over PM windows which are children of this facet's facet window are handled by this facet's part in its [HandleEvent](#) method.

-----

## GetHandleMouseEvents (OS/2) - Parameters

**rv** ([ODULong](#)) - returns

A bit field which can contain one or both of the following flags ORed together.

kODPartHandlesDragDropEvents

Drag-and-drop events which occur over PM windows which are children of this facet's facet window are handled by this facet's part in its [DragEnter](#), [DragWithin](#), and [DragLeave](#) methods.

kODPartHandlesMouseEvents

Mouse events (excluding drag-and-drop events) which occur over PM windows which are children of this facet's facet window are handled by this facet's part in its [HandleEvent](#) method.

-----

## GetHandleMouseEvents (OS/2) - Remarks

The OpenDoc standard dispatch module calls this method to determine how to dispatch events that occur over a facet which contains non-OpenDoc PM windows.

-----

## GetHandleMouseEvents (OS/2) - Related Methods

### Related Methods

- [ODFacet::GetFacetHWND](#)
- [ODFacet::SetHandleMouseEvents](#)

-----

## GetHandleMouseEvents (OS/2) - Topics

### Class:

ODFacet

Select an item:

[Syntax](#)

[Parameters](#)

---

# GetHighlight

---

## GetHighlight - Syntax

This method returns the highlight state for this facet.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODHighlight    rv;

rv = GetHighlight();
```

---

## GetHighlight Return Value - rv

**rv** ([ODHighlight](#)) - returns

The highlight state for this facet. This parameter can be set to one of the following values:

kODDimHighlight	The facet is highlighted in background style.
kODFullHighlight	The facet is highlighted in foreground style.
kODNoHighlight	The facet is not highlighted.

---

## GetHighlight - Parameters

**rv** ([ODHighlight](#)) - returns

The highlight state for this facet. This parameter can be set to one of the following values:

kODDimHighlight	The facet is highlighted in background style.
kODFullHighlight	The facet is highlighted in foreground style.
kODNoHighlight	The facet is not highlighted.

---

## GetHighlight - Remarks



This facet's part uses the highlight state to draw its content consistently with the content highlighting of this facet's containing part.

---

## GetHighlight - Related Methods

### Related Methods

- [ODFacet::ChangeHighlight](#)
- 

## GetHighlight - Topics

### Class:

[ODFacet](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## GetPartInfo

---

## GetPartInfo - Syntax

This method returns the part-information data for this facet.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODInfoType    rv;

rv = GetPartInfo();
```

---

## GetPartInfo Return Value - rv

**rv** ([ODInfoType](#)) - returns  
The part-information data for this facet.

---

## GetPartInfo - Parameters

**rv** ([ODInfoType](#)) - returns  
The part-information data for this facet.

---

## GetPartInfo - Remarks

You should cast the return value to a pointer to your part's own representation of the data.

---

## GetPartInfo - Related Methods

### Related Methods

- [ODFacet::SetPartInfo](#)
  - [ODFrame::GetPartInfo](#)
- 

## GetPartInfo - Topics

**Class:**  
[ODFacet](#)

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## GetWindow

---

## GetWindow - Syntax

This method returns a reference to the window in which this facet is displayed.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>
```

```
ODWindow      *rv;  
  
rv = GetWindow();
```

---

## GetWindow Return Value - rv

**rv** (ODWindow \*) - returns

A reference to the window in which this facet is displayed or KODNULL if this facet does not actually appear in any window (for example, a printing facet does not appear in a window).

---

## GetWindow - Parameters

**rv** (ODWindow \*) - returns

A reference to the window in which this facet is displayed or KODNULL if this facet does not actually appear in any window (for example, a printing facet does not appear in a window).

---

## GetWindow - Remarks

If this facet has no window of its own, this method searches recursively for the containing facet's window. Only the root facet of a window has a reference to the window; all its embedded facets then inherit this value.

---

## GetWindow - Related Methods

### Related Methods

- [ODFrame::AcquireWindow](#)

---

## GetWindow - Topics

### Class:

ODFacet

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

# HasCanvas

---

## HasCanvas - Syntax

This method indicates whether this facet has its own canvas.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODBoolean    rv;

rv = HasCanvas();
```

---

## HasCanvas Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating whether this facet has its own canvas.

kODTrue	The facet has its own canvas.
kODFalse	The facet does not have its own canvas.

---

## HasCanvas - Parameters

**rv** ([ODBoolean](#)) - returns

A flag indicating whether this facet has its own canvas.

kODTrue	The facet has its own canvas.
kODFalse	The facet does not have its own canvas.

---

## HasCanvas - Related Methods

### Related Methods

- [ODFacet::ChangeCanvas](#)
- [ODFacet::GetCanvas](#)

---

# HasCanvas - Topics

## Class:

ODFacet

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Related Methods](#)

---

## Invalidate

---

### Invalidate - Syntax

This method marks the specified area in this facet as in need of updating.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODShape      *invalidShape;
ODCanvas     *biasCanvas;

Invalidate(invalidShape, biasCanvas);
```

---

### Invalidate Parameter - invalidShape

**invalidShape** (ODShape \*) - input

A reference to the shape object defining the area in this facet that needs updating, expressed in frame coordinates.

---

### Invalidate Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in platform-normal coordinates.

---

### Invalidate - Return Value

None.

-----

## Invalidate - Parameters

**invalidShape** (ODShape \*) - input

A reference to the shape object defining the area in this facet that needs updating, expressed in frame coordinates.

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in platform-normal coordinates.

None.

-----

## Invalidate - Remarks

OpenDoc calls this method. This method transforms and clips the shape from the coordinate space of the facet to the coordinate space its canvas. The shape is marked on the canvas (and all parent canvases) associated with this facet, ensuring that changes to this facet are reflected across all appropriate canvases.

-----

## Invalidate - Exception Handling

kODErrInvalidFacet

The facet associated with the specified canvas does not have a parent facet.

-----

## Invalidate - Related Methods

### Related Methods

- [ODCanvas::Invalidate](#)
- [ODFacet::Validate](#)
- [ODFrame::Invalidate](#)

-----

## Invalidate - Topics

### Class:

ODFacet

Select an item:

---

# InvalidateActiveBorder

---

## InvalidateActiveBorder - Syntax

This method marks the active frame border of this facet as in need of updating.

```
#define INCL_ODFACET  
#define INCL_ODAPI  
#include <os2.h>
```

```
InvalidateActiveBorder();
```

---

## InvalidateActiveBorder - Return Value

None.

---

## InvalidateActiveBorder - Parameters

None.

---

## InvalidateActiveBorder - Remarks

OpenDoc calls this method when invalidating the active frame border if there is no explicit active frame border defined. When this facet's frame has the selection focus, OpenDoc automatically takes care of changing the facet's active border shape and redrawing the active frame border.

Under normal circumstances, there is no need to invalidate the active frame border for either the active part or its containing part; however, parts could call this method if they want to force the active frame border to be redrawn when they change the shape of an embedded part. To do so, the part must invalidate the active frame border by calling its facet's `InvalidateActiveBorder` method and redraw the active frame border by calling its facet's [DrawActiveBorder](#) method.

---

# InvalidateActiveBorder - Related Methods

## Related Methods

- [ODFacet::DrawActiveBorder](#)
  - [ODFrame::InvalidateActiveBorder](#)
- 

# InvalidateActiveBorder - Topics

## Class:

[ODFacet](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

# IsSelected

---

## IsSelected - Syntax

This method indicates whether this facet is selected within its containing part.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    rv;
```

```
rv = IsSelected();
```

---

## IsSelected Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating whether this facet is selected within its containing part.

kODTrue

This facet is selected within the containing part.

kODFalse

This facet is not selected within the containing part.



---

## IsSelected - Parameters

**rv** ([ODBoolean](#)) - returns

A flag indicating whether this facet is selected within its containing part.

kODTrue

This facet is selected within the containing part.

kODFalse

This facet is not selected within the containing part.

---

## IsSelected - Remarks

A containing part calls this method when it believes this facet's frame has the selection focus. The criteria for facet selection is specific to the containing part which calls this method. OpenDoc uses the return value to determine whether to dispatch mouse events to this facet or to its containing facet.

---

## IsSelected - Related Methods

### Related Methods

- [ODFacet::SetSelected](#)
- 

## IsSelected - Topics

### Class:

[ODFacet](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

## MoveBefore

---

## MoveBefore - Syntax

This method repositions an embedded facet of this facet in front of a sibling facet.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODFacet      *child;
ODFacet      *sibling;

MoveBefore(child, sibling);
```

---

## MoveBefore Parameter - child

**child** (ODFacet \*) - input  
A reference to a embedded facet to be repositioned.

---

## MoveBefore Parameter - sibling

**sibling** (ODFacet \*) - input  
A reference to a sibling facet to be directly behind the embedded facet or KODNULL if the embedded facet is to be repositioned in front of all its sibling facets.

---

## MoveBefore - Return Value

None.

---

## MoveBefore - Parameters

**child** (ODFacet \*) - input  
A reference to a embedded facet to be repositioned.

**sibling** (ODFacet \*) - input  
A reference to a sibling facet to be directly behind the embedded facet or KODNULL if the embedded facet is to be repositioned in front of all its sibling facets.

None.

---

## MoveBefore - Remarks

After this method executes successfully, the sibling order associated with the embedded facets may or may not have changed. Any changes to the sibling order of the embedded facets are reflected in the facet iterator, if a facet iterator exists.

While you are using a facet iterator, you should not call this method to modify the list of embedded facets. You must postpone repositioning facets in this list of embedded facets until you have deleted the iterator.

---

## MoveBefore - Exception Handling

kODErrInvalidFacet

The child or sibling facet is not an embedded facet of this facet.

---

## MoveBefore - Related Methods

### Related Methods

- [ODFacet::MoveBehind](#)

---

## MoveBefore - Topics

### Class:

ODFacet

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## MoveBehind

---

## MoveBehind - Syntax

This method repositions an embedded facet of this facet behind a sibling facet.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>
```

```
ODFacet    *child;
ODFacet    *sibling;
```

```
MoveBehind(child, sibling);
```

---

## MoveBehind Parameter - child

**child** (ODFacet \*) - input  
A reference to a embedded facet to be repositioned.

---

## MoveBehind Parameter - sibling

**sibling** (ODFacet \*) - input  
A reference to a sibling facet to be moved directly in front of the embedded facet or KODNULL if the embedded facet is repositioned behind all its sibling facets.

---

## MoveBehind - Return Value

None.

---

## MoveBehind - Parameters

**child** (ODFacet \*) - input  
A reference to a embedded facet to be repositioned.

**sibling** (ODFacet \*) - input  
A reference to a sibling facet to be moved directly in front of the embedded facet or KODNULL if the embedded facet is repositioned behind all its sibling facets.

None.

---

## MoveBehind - Remarks

After this method executes successfully, the sibling order associated with the embedded facets may or may not have changed. Any changes to the sibling order of the embedded facets are reflected in the facet iterator, if a facet iterator exists.

While you are using a facet iterator, you should not call this method to modify the list of embedded facets. You must postpone repositioning facets in this list of embedded facets until after you have deleted the iterator.

---

# MoveBehind - Exception Handling

kODErrInvalidFacet

The child or sibling facet is not an embedded facet of this facet.

---

## MoveBehind - Related Methods

### Related Methods

- [ODFacet::MoveBefore](#)

---

## MoveBehind - Topics

### Class:

ODFacet

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## RemoveFacet

---

## RemoveFacet - Syntax

This method removes an embedded facet from this facet.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>
```

```
ODFacet      *facet;
```

```
RemoveFacet (facet);
```

---

## RemoveFacet Parameter - facet

**facet** (ODFacet \*) - input  
A reference to an embedded facet to be removed.

---

## RemoveFacet - Return Value

None.

---

## RemoveFacet - Parameters

**facet** (ODFacet \*) - input  
A reference to an embedded facet to be removed.

None.

---

## RemoveFacet - Remarks

This facet's containing part calls this method before removing one of its embedded frames or optionally when scrolling an embedded frame out of view. This method in turn calls the [FacetRemoved](#) method associated with this facet's part to notify the part that a facet has been removed from one of its frames. After this method executes successfully, the removed facet should not be used again; the caller should delete the removed facet.

---

## RemoveFacet - Exception Handling

kODErrInvalidFacet

The specified facet is not an embedded facet of this facet.

---

## RemoveFacet - Related Methods

### Related Methods

- [ODFrame::FacetRemoved](#)
  - [ODPart::FacetRemoved](#)
- 

## RemoveFacet - Topics

**Class:**

ODFacet

Select an item:

[Syntax](#)[Parameters](#)[Returns](#)[Remarks](#)[Exception Handling](#)[Related Methods](#)

---

## SetHandleMouseEvents (OS/2)

---

### SetHandleMouseEvents (OS/2) - Syntax

This method indicates whether events that occur over non-OpenDoc PM windows, such as buttons or scroll bars, which are children of this facet's facet window are sent to the PM window or to this facet's part in its [HandleEvent](#) method.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODULong    flags;

SetHandleMouseEvents(flags);
```

---

### SetHandleMouseEvents (OS/2) Parameter - flags

**flags** ([ODULong](#)) - input

A bit field which can contain one or both of the following flags ORed together:

kODPartHandlesDragDropEvents

Drag-and-drop events which occur over PM windows which are children of this facet's facet window are to be handled by this facet's part in its [DragEnter](#), [DragWithin](#), and [DragLeave](#) methods.

kODPartHandlesMouseEvents

Mouse events (excluding drag-and-drop events) which occur over PM windows which are children of this facet's facet window are to be handled by this facet's part in its [HandleEvent](#) method.

---

### SetHandleMouseEvents (OS/2) - Return Value

None.

---

## SetHandleMouseEvents (OS/2) - Parameters

**flags** ([ODULong](#)) - input

A bit field which can contain one or both of the following flags ORed together:

kODPartHandlesDragDropEvents

Drag-and-drop events which occur over PM windows which are children of this facet's facet window are to be handled by this facet's part in its [DragEnter](#), [DragWithin](#), and [DragLeave](#) methods.

kODPartHandlesMouseEvents

Mouse events (excluding drag-and-drop events) which occur over PM windows which are children of this facet's facet window are to be handled by this facet's part in its [HandleEvent](#) method.

None.

---

## SetHandleMouseEvents (OS/2) - Remarks

By default, all mouse and drag-and-drop events which occur over PM windows are sent to that window.

---

## SetHandleMouseEvents (OS/2) - Related Methods

### Related Methods

- [ODFacet::GetFacetHWND](#)
  - [ODFacet::GetHandleMouseEvents](#)
- 

## SetHandleMouseEvents (OS/2) - Topics

### Class:

[ODFacet](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## SetPartInfo

---



# SetPartInfo - Syntax

This method assigns part-information data to this facet.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODInfoType    partInfo;

SetPartInfo(partInfo);
```

---

## SetPartInfo Parameter - partInfo

**partInfo** ([ODInfoType](#)) - input  
The data for this facet's part information.

---

## SetPartInfo - Return Value

None.

---

## SetPartInfo - Parameters

**partInfo** ([ODInfoType](#)) - input  
The data for this facet's part information.

None.

---

## SetPartInfo - Related Methods

### Related Methods

- [ODFacet::GetPartInfo](#)
  - [ODFrame::SetPartInfo](#)
- 

## SetPartInfo - Topics

**Class:**

ODFacet

Select an item:

[Syntax](#)[Parameters](#)[Returns](#)[Related Methods](#)

---

## SetSelected

---

## SetSelected - Syntax

This method specifies whether this facet is currently being selected within its containing part.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODBoolean    isSelected;

SetSelected(isSelected);
```

---

## SetSelected Parameter - isSelected

**isSelected** ([ODBoolean](#)) - input

A flag indicating whether this facet is currently being selected within its containing part.

kODTrue	This facet is currently being selected.
kODFalse	This facet is not currently being selected.

---

## SetSelected - Return Value

None.

---

## SetSelected - Parameters

**isSelected** ([ODBoolean](#)) - input

A flag indicating whether this facet is currently being selected within its containing part.

kODTrue

This facet is currently being selected.

kODFalse

This facet is not currently being selected.

None.

---

## SetSelected - Remarks

The criteria for facet selection is specific to the containing part that calls the method.

---

## SetSelected - Related Methods

### Related Methods

- [ODFacet::isSelected](#)
- 

## SetSelected - Topics

### Class:

ODFacet

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

## Update

---

## Update - Syntax

This method updates this facet's canvas by drawing this facet and any of its embedded facets whose clip shape intersects the specified area of the canvas.

```
#define INCL_ODFACET
```

```
#define INCL_ODAPI
#include <os2.h>

ODShape      *invalidShape;
ODCanvas     *biasCanvas;

Update(invalidShape, biasCanvas);
```

-----

## Update Parameter - invalidShape

**invalidShape** (ODShape \*) - input

A reference to a shape object defining the area of the canvas that needs updating, expressed in frame coordinates or KODNULL if the facets are to be updated based on the invalid shape for the facet's canvas.

-----

## Update Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates.

-----

## Update - Return Value

None.

-----

## Update - Parameters

**invalidShape** (ODShape \*) - input

A reference to a shape object defining the area of the canvas that needs updating, expressed in frame coordinates or KODNULL if the facets are to be updated based on the invalid shape for the facet's canvas.

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates.

None.

-----

## Update - Remarks

When an update event occurs that involves a facet of your part, OpenDoc calls its window's [Update](#) method, which in turn calls this method.

This method in turn call the [Draw](#) method associated with this facet's part.

On OS/2, if you want to control the sequence of updating embedded parts from within your part's [Draw](#) method (to interleave drawing of embedded parts with content objects, for example), you can call the Update method of an embedded facet with the *invalidShape* parameter equal to KODNULL. This causes the embedded facet (and its embedded facets, and so on) to be updated as required. When KODNULL is specified for the *invalidShape* parameter, facets are updated based on the invalid shape for the facet's canvas. If you modify the invalid shape of an embedded facet from within your draw method prior to calling that facet's Update method, you should specify the modified invalid shape (expressed in the embedded part's frame coordinates).

---

## Update - Related Methods

### Related Methods

- [ODPart::Draw](#)
- [ODWindow::Update](#)

---

## Update - Topics

### Class:

ODFacet

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## Validate

---

## Validate - Syntax

This method marks the specified area in this facet as no longer in need of updating.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODShape      *validShape;
ODCanvas     *biasCanvas;

Validate(validShape, biasCanvas);
```

---

## Validate Parameter - validShape

**validShape** (ODShape \*) - input

A reference to a shape object defining the area in this facet that no longer needs updating, expressed in frame coordinates.

---

## Validate Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates.

---

## Validate - Return Value

None.

---

## Validate - Parameters

**validShape** (ODShape \*) - input

A reference to a shape object defining the area in this facet that no longer needs updating, expressed in frame coordinates.

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates.

None.

---

## Validate - Remarks

OpenDoc calls this method. This method transforms and clips the shape from the coordinate space of the facet to the coordinate space of its canvas. It then subtracts the shape from any existing invalid area of the canvas.

---

## Validate - Related Methods

### Related Methods

- [ODCanvas::Validate](#)
- [ODFacet::Invalidate](#)
- [ODFrame::Validate](#)

---

# Validate - Topics

## Class:

ODFacet

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

## ODFacetIterator

**Class Definition File:** FACETITR.IDL

### Class Hierarchy

SOMObject

ODObject

**ODFacetIterator**

### Description

An object of the ODFacetIterator class provides access to all facets embedded in a facet.

A facet iterator is used to apply an operation to a facet and all its embedded facets. For example, you might use a facet iterator to make sure that all facets within a facet being invalidated are also invalidated.

Your part creates a facet iterator object by calling a facet's [CreateFacetIterator](#) method which returns a reference to a facet iterator object.

When you create a facet iterator, you specify the traversal type and sibling order. These two characteristics determine which facets are included in the iteration sequence and the order of the facets within the iteration sequence.

- If you specify top-down traversal, traverse the facet hierarchy top down, in depth-first order.
- If you specify bottom-up traversal, traverse the facet hierarchy bottom up, visiting the specified facet after visiting all its children. If the sibling order is front-to-back, the traversal starts with the front-most facet at the lowest level in the hierarchy. If the sibling order is back-to-front, the traversal starts with the back-most facet at the lowest level in the hierarchy.
- If you specify children-only traversal, traverse only the children of the specified facet (not including the specified facet itself).

While you are using a facet iterator, you should not modify the list of embedded facets or call the facet's [MoveBefore](#) or [MoveBehind](#) method. You must postpone adding facets to or removing facets from the list of embedded facets until after you have deleted the iterator.

For more information related to facet objects, see the class description for [ODFacet](#). For more information on accessing objects through iterators, see the chapter on OpenDoc run-time features in the *OpenDoc Programming Guide* .

### Methods

The methods defined by the ODFacetIterator class include:

- [First](#)
- [IsNotComplete](#)
- [Next](#)
- [SkipChildren](#)

### Overridden Methods

There are currently no methods overridden by the ODFacetIterator class.

---

## First

---

## First - Syntax

This method begins the iteration and returns a reference to the first facet in the iteration sequence, as indicated by the traversal type and sibling order that were set for this iterator.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODFacet      *rv;

rv = First();
```

---

## First Return Value - rv

**rv** (ODFacet \*) - returns

A reference to the first facet in the iteration sequence or KODNULL if the traversal type is children-only and the root facet has no embedded facets.

---

## First - Parameters

**rv** (ODFacet \*) - returns

A reference to the first facet in the iteration sequence or KODNULL if the traversal type is children-only and the root facet has no embedded facets.

---

## First - Remarks

The traversal type and sibling order are set by the facet's [CreateFacetIterator](#) method.

Your part must call this method before calling this facet iterator's [IsNotComplete](#) method for the first time. This method may be called multiple times; each time, it resets the iteration.

---

## First - Exception Handling

KODErrIteratorOutOfSync

The list of embedded facets was modified while the iteration was in progress.

---



# First - Related Methods

## Related Methods

- [ODFacet::CreateFacetIterator](#)

# First - Topics

## Class:

[ODFacetIterator](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

# IsNotComplete

## IsNotComplete - Syntax

This method indicates whether the iteration is incomplete.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    rv;
```

```
rv = IsNotComplete();
```

## IsNotComplete Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the iteration is incomplete.

TRUE

The iteration is incomplete.

FALSE

The iteration is complete.

---

## IsNotComplete - Parameters

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the iteration is incomplete.

TRUE

The iteration is incomplete.

FALSE

The iteration is complete.

---

## IsNotComplete - Remarks

Your part calls this method to test whether more facets remain in the iteration sequence. This method returns kODTrue if the preceding call to the [First](#) or [Next](#) method found a facet. This method returns kODFalse when you have examined all the facets (that is, when the previous call to [First](#) or [Next](#) returned kODNULL).

---

## IsNotComplete - Exception Handling

kODErrIteratorNotInitialized

This method was called before calling either the [First](#) or [Next](#) method to begin the iteration.

kODErrIteratorOutOfSync

The list of embedded facets was modified while the iteration was in progress.

---

## IsNotComplete - Topics

### Class:

[ODFacetIterator](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

---

## Next

---

## Next - Syntax

This method returns a reference to the next facet in the iteration sequence.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODFacet      *rv;

rv = Next();
```

---

## Next Return Value - rv

**rv** (ODFacet \*) - returns

A reference to the next facet in the iteration sequence or kODNULL if you have reached the last facet.

---

## Next - Parameters

**rv** (ODFacet \*) - returns

A reference to the next facet in the iteration sequence or kODNULL if you have reached the last facet.

---

## Next - Remarks

If you part calls this method before calling this facet iterator's [First](#) method to begin the iteration, then this method works the same as calling the [First](#) method.

---

## Next - Exception Handling

kODErrIteratorOutOfSync

The list of embedded facets was modified while the iteration was in progress.

---

## Next - Topics

**Class:**

ODFacetIterator

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

## SkipChildren

---

## SkipChildren - Syntax

This method advances to the next sibling, skipping over the embedded facets of the current facet if the traversal type of this iteration is top-down.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>
```

```
SkipChildren();
```

---

## SkipChildren - Return Value

None.

---

## SkipChildren - Parameters

None.

---

## SkipChildren - Remarks

If the traversal type for this iterator is not top-down, calling this method has no effect.

---

## SkipChildren - Topics

**Class:**  
ODFacetIterator

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

# ODFocusModule

**Class Definition File:** FOCUSMOD.IDL

## Class Hierarchy

SOMObject  
  ODObject  
    **ODFocusModule**

## Description

An object of the ODFocusModule class is used to manage particular types of focus or ownership of a shared resource.

A focus is a designation of ownership of a given shared resource or feature, such as the keyboard or menu bar. A frame that owns an event-related focus receives events pertaining to that resource. A *focus module* manages a particular focus, maintaining the identity of the individual frame that own the focus. The arbitrator uses at least one internal focus module to handle standard OpenDoc event types of a particular platform. Typically, you do not need to subclass ODFocusModule or even access the internal focus module directly; however, you can define additional focus types, as needed, to handle other kinds of user events (such as input from new kinds of devices) by creating a new kind of focus module.

The ODFocusModule class is an abstract superclass that you can subclass to create a focus module. You can create a focus module object from within a shell plug-in object or from within one of your part's methods that are called during startup.

Before OpenDoc is able to recognize your new type of focus, you must register the associated focus module with the arbitrator. To register a focus module, you call the arbitrator's [RegisterFocus](#) method; to remove the focus module, you can call the arbitrator's [UnregisterFocus](#) method.

For more information related to the arbitrator, see the class description for [ODArbitrator](#). For more information on creating custom focus types, see the chapter on extending OpenDoc in the *OpenDoc Programming Guide*.

## Overriding Inherited Methods

The following methods are inherited and available for use by your subclass of ODFocusModule.

### somInit

The somInit method initializes the instance variables in a SOM object; it is inherited from the SOMObject class.

If you subclass ODFocusModule, you can override this method. Your override method does not need to call its inherited method; the inherited method is automatically called for you by the SOM library.

Your override of this method should initialize the new instance variables in this focus-module object. The SOM library calls this method when this focus module is created. You must not do anything that might cause this method to fail. This limits you to operations like setting pointer variables to null, setting numeric variables to appropriate values, and making similar assignments from constants. If you have any initialization code that can potentially fail, it must be handled in this focus module's subclass-specific initialization method; see also the [InitFocusModule](#) method.

### somUninit

The somUninit method disposes of the storage created for a SOM object; it is inherited from the SOMObject class.

If you subclass ODFocusModule, you can override this method. Your override method does not need to call its inherited method; the inherited method is automatically called for you by the SOM library.

Your override method should dispose of any storage created for this focus-module object, including any storage related to additional instance variables initialized in this focus-module object. The SOM library calls this method when this focus module is deleted; this method must not fail.

## Methods

The methods defined by the ODFocusModule class include:

- [AbortRelinquishFocus](#)
- [AcquireFocusOwner](#)

- [BeginRelinquishFocus](#)
- [CommitRelinquishFocus](#)
- [CreateOwnerIterator](#)
- [InitFocusModule](#)
- [IsFocusExclusive](#)
- [SetFocusOwnership](#)
- [TransferFocusOwnership](#)
- [UnsetFocusOwnership](#)

## Overridden Methods

There are currently no methods overridden by the ODFocusModule class.

# AbortRelinquishFocus

## AbortRelinquishFocus - Syntax

This method should cancel the request for the frame to relinquish ownership of the specified exclusive focus.

```
#define INCL_ODFOCUSMODULE
#define INCL_ODAPI
#include <os2.h>

ODTypeToken    focus;
ODFrame        *requestingFrame;
```

```
AbortRelinquishFocus(focus, requestingFrame);
```

## AbortRelinquishFocus Parameter - focus

**focus** ([ODTypeToken](#)) - input

A tokenized string representing the focus type which was to be relinquished.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus	The frame with the clipboard focus has access to the clipboard.
kODKeyFocus	The frame with keyboard focus receives keyboard events (excluding page-movement key events).
kODMenuFocus	The frame with menu focus receives menu events.
kODModalFocus	A frame with modal focus is notifying other frames that it is the only currently modal frame.
kODMouseFocus	The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.
kODScrollingFocus	The frame with the scrolling focus receives page-movement key events (such as <b>PageUp</b> ). This frame does not need to own keyboard focus.
kODSelectionFocus	The frame with the selection focus receives modified mouse-click events ( <b>Shift</b> +click and <b>Ctrl</b> +click). OpenDoc draws the active frame border around all facets of this frame.

---

# AbortRelinquishFocus Parameter - requestingFrame

**requestingFrame** (ODFrame \*) - input  
A reference to a frame that originally requested the focus.

---

## AbortRelinquishFocus - Return Value

None.

---

## AbortRelinquishFocus - Parameters

**focus** (ODTypeToken) - input  
A tokenized string representing the focus type which was to be relinquished.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus	The frame with the clipboard focus has access to the clipboard.
kODKeyFocus	The frame with keyboard focus receives keyboard events (excluding page-movement key events).
kODMenuFocus	The frame with menu focus receives menu events.
kODModalFocus	A frame with modal focus is notifying other frames that it is the only currently modal frame.
kODMouseFocus	The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.
kODScrollingFocus	The frame with the scrolling focus receives page-movement key events (such as <b>PageUp</b> ). This frame does not need to own keyboard focus.
kODSelectionFocus	The frame with the selection focus receives modified mouse-click events ( <b>Shift</b> +click and <b>Ctrl</b> +click). OpenDoc draws the active frame border around all facets of this frame.

**requestingFrame** (ODFrame \*) - input  
A reference to a frame that originally requested the focus.

None.

---

## AbortRelinquishFocus - Remarks

OpenDoc calls this method, which in turns calls the [AbortRelinquishFocus](#) method of the part that owns the focus. This method should give those focus owners who have indicated willingness to relinquish focus an opportunity to back out of changes initiated when OpenDoc first called the part's [BeginRelinquishFocus](#) method.

---

# AbortRelinquishFocus - Override Policy

If you subclass [ODFocusModule](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## AbortRelinquishFocus - Related Methods

### Related Methods

- [ODFocusModule::BeginRelinquishFocus](#)
  - [ODFocusModule::CommitRelinquishFocus](#)
  - [ODPart::AbortRelinquishFocus](#)
  - [ODPart::BeginRelinquishFocus](#)
  - [ODSession::Tokenize](#)
- 

## AbortRelinquishFocus - Topics

### Class:

[ODFocusModule](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Related Methods](#)

---

## AcquireFocusOwner

---

## AcquireFocusOwner - Syntax

This method should return a reference to the frame that owns the specified exclusive focus.

```
#define INCL_ODFOCUSMODULE
#define INCL_ODAPI
#include <os2.h>

ODTypeToken    focus;
ODFrame        *rv;

rv = AcquireFocusOwner(focus);
```

---



# AcquireFocusOwner Parameter - focus

**focus** (ODTypeToken) - input

A tokenized string representing the focus type whose owner is desired.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the [clipboard](#) focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus

The frame with menu focus receives menu events.

kODModalFocus

A frame with modal focus is notifying other frames that it is the only currently modal frame.

kODMouseFocus

The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

kODScrollingFocus

The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

kODSelectionFocus

The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

-----

# AcquireFocusOwner Return Value - rv

**rv** (ODFrame \*) - returns

A reference to the frame that owns the specified exclusive focus or kODNULL if the focus is not owned by any frame.

-----

# AcquireFocusOwner - Parameters

**focus** (ODTypeToken) - input

A tokenized string representing the focus type whose owner is desired.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the [clipboard](#) focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus

The frame with menu focus receives menu events.

kODModalFocus

A frame with modal focus is notifying other frames that it is the only currently modal frame.

kODMouseFocus

The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

kODScrollingFocus

The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

kODSelectionFocus

The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

**rv** (ODFrame \*) - returns

A reference to the frame that owns the specified exclusive focus or kODNULL if the focus is not owned by any frame.

---

## AcquireFocusOwner - Remarks

OpenDoc calls this method. A part can obtain a reference to the owner of a specified exclusive focus by calling the arbitrator's [AcquireFocusOwner](#) method, which in turn calls this method. If the focus is not registered, the focus has no focus module and this method is never called.

Before returning the frame object, your override method should call the frame object's [Acquire](#) method. When the caller has finished using the returned frame object, it should call the frame object's [Release](#) method.

---

## AcquireFocusOwner - Override Policy

If you subclass [ODFocusModule](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## AcquireFocusOwner - Related Methods

### Related Methods

- [ODArbitrator::AcquireFocusOwner](#)
  - [ODSession::Tokenize](#)
- 

## AcquireFocusOwner - Topics

### Class:

[ODFocusModule](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Related Methods](#)

---

## BeginRelinquishFocus

---

## BeginRelinquishFocus - Syntax

This method should indicate whether the current owner of the specified exclusive focus is willing to give up ownership of the focus.

```
#define INCL_ODFOCUSMODULE
#define INCL_ODAPI
#include <os2.h>

ODTypeToken    focus;
ODFrame        *requestingFrame;
ODBoolean      rv;

rv = BeginRelinquishFocus(focus, requestingFrame);
```

---

## BeginRelinquishFocus Parameter - focus

**focus** (ODTypeToken) - input

A tokenized string representing the focus type to be relinquished.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus	The frame with the clipboard focus has access to the clipboard.
kODKeyFocus	The frame with keyboard focus receives keyboard events (excluding page-movement key events).
kODMenuFocus	The frame with menu focus receives menu events.
kODModalFocus	A frame with modal focus is notifying other frames that it is the only currently modal frame.
kODMouseFocus	The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.
kODScrollingFocus	The frame with the scrolling focus receives page-movement key events (such as <b>PageUp</b> ). This frame does not need to own keyboard focus.
kODSelectionFocus	The frame with the selection focus receives modified mouse-click events ( <b>Shift</b> +click and <b>Ctrl</b> +click). OpenDoc draws the active frame border around all facets of this frame.

---

## BeginRelinquishFocus Parameter - requestingFrame

**requestingFrame** (ODFrame \*) - input

A reference to the frame requesting ownership of the focus.

---

## BeginRelinquishFocus Return Value - rv

**rv** (ODBoolean) - returns

A flag indicating whether the current owner of the specified exclusive focus is willing to give up ownership of the focus.

kODTrue	The current owner is willing to give up ownership.
kODFalse	

The current owner is not willing to give up ownership.

---

## BeginRelinquishFocus - Parameters

**focus** ([ODTypeToken](#)) - input

A tokenized string representing the focus type to be relinquished.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the clipboard focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus

The frame with menu focus receives menu events.

kODModalFocus

A frame with modal focus is notifying other frames that it is the only currently modal frame.

kODMouseFocus

The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

kODScrollingFocus

The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

kODSelectionFocus

The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

**requestingFrame** ([ODFrame \\*](#)) - input

A reference to the frame requesting ownership of the focus.

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the current owner of the specified exclusive focus is willing to give up ownership of the focus.

kODTrue

The current owner is willing to give up ownership.

kODFalse

The current owner is not willing to give up ownership.

---

## BeginRelinquishFocus - Remarks

OpenDoc calls this method, which in turn calls the [BeginRelinquishFocus](#) method of the part that owns the focus. If the part's [BeginRelinquishFocus](#) method returns kODTrue (the typical case), this method should return kODTrue; if the part's [BeginRelinquishFocus](#) method returns kODFalse, this method should return kODFalse.

---

## BeginRelinquishFocus - Override Policy

If you subclass [ODFocusModule](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## BeginRelinquishFocus - Related Methods

### Related Methods

- [ODFocusModule::AbortRelinquishFocus](#)
- [ODFocusModule::CommitRelinquishFocus](#)
- [ODPart::BeginRelinquishFocus](#)
- [ODSession::Tokenize](#)

---

## BeginRelinquishFocus - Topics

### Class:

[ODFocusModule](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Related Methods](#)

---

## CommitRelinquishFocus

---

## CommitRelinquishFocus - Syntax

This method should signal to the part that owns the specified exclusive focus that it is about to lose the ownership of the it.

```
#define INCL_ODFOCUSMODULE
#define INCL_ODAPI
#include <os2.h>

ODTypeToken    focus;
ODFrame        *requestingFrame;

CommitRelinquishFocus(focus, requestingFrame);
```

---

## CommitRelinquishFocus Parameter - focus

**focus** ([ODTypeToken](#)) - input

A tokenized string representing the focus type to be relinquished.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus	The frame with the clipboard focus has access to the clipboard.
kODKeyFocus	The frame with keyboard focus receives keyboard events (excluding page-movement key events).
kODMenuFocus	

	The frame with menu focus receives menu events.
kODModalFocus	A frame with modal focus is notifying other frames that it is the only currently modal frame.
kODMouseFocus	The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.
kODScrollingFocus	The frame with the scrolling focus receives page-movement key events (such as <b>PageUp</b> ). This frame does not need to own keyboard focus.
kODSelectionFocus	The frame with the selection focus receives modified mouse-click events ( <b>Shift</b> +click and <b>Ctrl</b> +click). OpenDoc draws the active frame border around all facets of this frame.

-----

## CommitRelinquishFocus Parameter - requestingFrame

**requestingFrame** (ODFrame \*) - input  
A reference to a frame that requested the focus.

-----

## CommitRelinquishFocus - Return Value

None.

-----

## CommitRelinquishFocus - Parameters

**focus** (ODTypeToken) - input  
A tokenized string representing the focus type to be relinquished.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus	The frame with the clipboard focus has access to the clipboard.
kODKeyFocus	The frame with keyboard focus receives keyboard events (excluding page-movement key events).
kODMenuFocus	The frame with menu focus receives menu events.
kODModalFocus	A frame with modal focus is notifying other frames that it is the only currently modal frame.
kODMouseFocus	The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.
kODScrollingFocus	The frame with the scrolling focus receives page-movement key events (such as <b>PageUp</b> ). This frame does not need to own keyboard focus.
kODSelectionFocus	The frame with the selection focus receives modified mouse-click events ( <b>Shift</b> +click and <b>Ctrl</b> +click). OpenDoc draws the active frame border around all facets of this frame.

**requestingFrame** (ODFrame \*) - input  
A reference to a frame that requested the focus.

None.

---

## CommitRelinquishFocus - Remarks

OpenDoc calls this method, which in turn calls the [CommitRelinquishFocus](#) method of the part that owns the focus.

---

## CommitRelinquishFocus - Override Policy

If you subclass [ODFocusModule](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## CommitRelinquishFocus - Related Methods

### Related Methods

- [ODFocusModule::AbortRelinquishFocus](#)
  - [ODFocusModule::BeginRelinquishFocus](#)
  - [ODPart::CommitRelinquishFocus](#)
  - [ODSession::Tokenize](#)
- 

## CommitRelinquishFocus - Topics

### Class:

[ODFocusModule](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Related Methods](#)

---

## CreateOwnerIterator

---

## CreateOwnerIterator - Syntax

This method should create a focus-owner iterator to give callers access to the frames that own the specified nonexclusive focus.

```
#define INCL_ODFOCUSMODULE  
#define INCL_ODAPI
```

```
#include <os2.h>

ODTypeToken          focus;
ODFocusOwnerIterator *rv;

rv = CreateOwnerIterator(focus);
```

---

## CreateOwnerIterator Parameter - focus

**focus** (ODTypeToken) - input

A tokenized string representing the focus type whose frames you want to enumerate.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus	The frame with the clipboard focus has access to the clipboard.
kODKeyFocus	The frame with keyboard focus receives keyboard events (excluding page-movement key events).
kODMenuFocus	The frame with menu focus receives menu events.
kODModalFocus	A frame with modal focus is notifying other frames that it is the only currently modal frame.
kODMouseFocus	The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.
kODScrollingFocus	The frame with the scrolling focus receives page-movement key events (such as <b>PageUp</b> ). This frame does not need to own keyboard focus.
kODSelectionFocus	The frame with the selection focus receives modified mouse-click events ( <b>Shift</b> +click and <b>Ctrl</b> +click). OpenDoc draws the active frame border around all facets of this frame.

---

## CreateOwnerIterator Return Value - rv

**rv** (ODFocusOwnerIterator \*) - returns

A reference to a new focus-owner iterator object or KODNULL if the focus is exclusive.

---

## CreateOwnerIterator - Parameters

**focus** (ODTypeToken) - input

A tokenized string representing the focus type whose frames you want to enumerate.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus	The frame with the clipboard focus has access to the clipboard.
kODKeyFocus	The frame with keyboard focus receives keyboard events (excluding page-movement key events).
kODMenuFocus	The frame with menu focus receives menu events.
kODModalFocus	A frame with modal focus is notifying other frames that it is the only currently modal frame.



kODMouseFocus

The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

kODScrollingFocus

The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

kODSelectionFocus

The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

**rv** (ODFocusOwnerIterator \*) - returns

A reference to a new focus-owner iterator object or kODNULL if the focus is exclusive.

-----

## CreateOwnerIterator - Remarks

OpenDoc calls this method. This method should create and initialize an instance of a focus-owner iterator that can iterate over this focus module's focus owners, and return the iterator to the caller.

While you are using the focus-owner iterator, you should not modify the list of focus owners. You must postpone adding items to or removing items from the list of facet owners until after you have deleted the iterator.

-----

## CreateOwnerIterator - Override Policy

If you subclass [ODFocusModule](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

-----

## CreateOwnerIterator - Related Methods

### Related Methods

- [ODSession::Tokenize](#)

-----

## CreateOwnerIterator - Topics

### Class:

[ODFocusModule](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Related Methods](#)

-----

## InitFocusModule

---

## InitFocusModule - Syntax

This method initializes this focus-module object.

```
#define INCL_ODFOCUSMODULE
#define INCL_ODAPI
#include <os2.h>

ODSession      *session;

InitFocusModule(session);
```

---

## InitFocusModule Parameter - session

**session** (ODSession \*) - input  
A reference to the current session object.

---

## InitFocusModule - Return Value

None.

---

## InitFocusModule - Parameters

**session** (ODSession \*) - input  
A reference to the current session object.

None.

---

## InitFocusModule - Remarks

This method is not called directly to initialize this focus-module object but is called by a subclass-specific initialization method. By convention, every subclass of [ODFocusModule](#) should have a separate initialization method (for example, the `InitMyFocusModule` method) that is called when an instance of that subclass is created. The override method may have additional parameters beyond those of the `InitFocusModule` method. The `InitMyFocusModule` method should call the inherited `InitFocusModule` method at the beginning of its implementation.

If you subclass [ODFocusModule](#) your subclass-specific initialization method, rather than its `somInit` method, should handle any initialization code that can potentially fail. For example, your initialization method may attempt to allocate memory for your focus module.

---

## InitFocusModule - Override Policy

If you subclass [ODFocusModule](#), you should not override this method.

---

## InitFocusModule - Topics

### Class:

[ODFocusModule](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

---

## IsFocusExclusive

---

## IsFocusExclusive - Syntax

This method should indicate whether the specified focus is exclusive.

```
#define INCL_ODFOCUSMODULE
#define INCL_ODAPI
#include <os2.h>

ODTypeToken    focus;
ODBoolean      rv;

rv = IsFocusExclusive(focus);
```

---

## IsFocusExclusive Parameter - focus

**focus** ([ODTypeToken](#)) - input

A tokenized string representing the focus type to be tested.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the [clipboard](#) focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus

	The frame with menu focus receives menu events.
kODModalFocus	A frame with modal focus is notifying other frames that it is the only currently modal frame.
kODMouseFocus	The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.
kODScrollingFocus	The frame with the scrolling focus receives page-movement key events (such as <b>PageUp</b> ). This frame does not need to own keyboard focus.
kODSelectionFocus	The frame with the selection focus receives modified mouse-click events ( <b>Shift</b> +click and <b>Ctrl</b> +click). OpenDoc draws the active frame border around all facets of this frame.

-----

## IsFocusExclusive Return Value - rv

**rv** ([ODBoolean](#)) - returns  
A flag indicating whether specified focus is exclusive.

kODTrue	The specified focus is exclusive.
kODFalse	The specified focus is not exclusive.

-----

## IsFocusExclusive - Parameters

**focus** ([ODTypeToken](#)) - input  
A tokenized string representing the focus type to be tested.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus	The frame with the clipboard focus has access to the clipboard.
kODKeyFocus	The frame with keyboard focus receives keyboard events (excluding page-movement key events).
kODMenuFocus	The frame with menu focus receives menu events.
kODModalFocus	A frame with modal focus is notifying other frames that it is the only currently modal frame.
kODMouseFocus	The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.
kODScrollingFocus	The frame with the scrolling focus receives page-movement key events (such as <b>PageUp</b> ). This frame does not need to own keyboard focus.
kODSelectionFocus	The frame with the selection focus receives modified mouse-click events ( <b>Shift</b> +click and <b>Ctrl</b> +click). OpenDoc draws the active frame border around all facets of this frame.

**rv** ([ODBoolean](#)) - returns  
A flag indicating whether specified focus is exclusive.

kODTrue	The specified focus is exclusive.
kODFalse	The specified focus is not exclusive.

-----

## IsFocusExclusive - Remarks

Foci may be exclusive or nonexclusive. All of the standard foci defined by OpenDoc are *exclusive* , meaning that only one frame can own the focus at a time. If you create a new kind of focus, you can make it *nonexclusive* , meaning that several frames could share ownership of it.

---

## IsFocusExclusive - Override Policy

If you subclass [ODFocusModule](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## IsFocusExclusive - Related Methods

### Related Methods

- [ODSession::Tokenize](#)
- 

## IsFocusExclusive - Topics

### Class:

[ODFocusModule](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Related Methods](#)

---

## SetFocusOwnership

---

## SetFocusOwnership - Syntax

This method should record the specified frame as the owner of the specified focus.

```
#define INCL_ODFOCUSMODULE
#define INCL_ODAPI
#include <os2.h>

ODTypeToken    focus;
ODFrame        *frame;

SetFocusOwnership(focus, frame);
```

---

## SetFocusOwnership Parameter - focus

**focus** (ODTypeToken) - input

A tokenized string representing the focus type whose owner is to be assigned.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the clipboard focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus

The frame with menu focus receives menu events.

kODModalFocus

A frame with modal focus is notifying other frames that it is the only currently modal frame.

kODMouseFocus

The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

kODScrollingFocus

The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

kODSelectionFocus

The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

---

## SetFocusOwnership Parameter - frame

**frame** (ODFrame \*) - input

A reference to a frame that is to own the focus.

---

## SetFocusOwnership - Return Value

None.

---

## SetFocusOwnership - Parameters

**focus** (ODTypeToken) - input

A tokenized string representing the focus type whose owner is to be assigned.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the clipboard focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus	The frame with menu focus receives menu events.
kODModalFocus	A frame with modal focus is notifying other frames that it is the only currently modal frame.
kODMouseFocus	The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.
kODScrollingFocus	The frame with the scrolling focus receives page-movement key events (such as <b>PageUp</b> ). This frame does not need to own keyboard focus.
kODSelectionFocus	The frame with the selection focus receives modified mouse-click events ( <b>Shift</b> +click and <b>Ctrl</b> +click). OpenDoc draws the active frame border around all facets of this frame.

**frame** (ODFrame \*) - input  
A reference to a frame that is to own the focus.

None.

---

## SetFocusOwnership - Remarks

OpenDoc calls this method. This method should record, in this focus module's internal structures, the new focus ownership.

---

## SetFocusOwnership - Override Policy

If you subclass [ODFocusModule](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## SetFocusOwnership - Related Methods

### Related Methods

- [ODSession::Tokenize](#)

---

## SetFocusOwnership - Topics

**Class:**  
[ODFocusModule](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Related Methods](#)

---

## TransferFocusOwnership

---

## TransferFocusOwnership - Syntax

This method should transfer ownership of the specified focus from one frame to another frame.

```
#define INCL_ODFOCUSMODULE
#define INCL_ODAPI
#include <os2.h>

ODTypeToken    focus;
ODFrame        *transferringFrame;
ODFrame        *newOwner;

TransferFocusOwnership(focus, transferringFrame,
                      newOwner);
```

---

## TransferFocusOwnership Parameter - focus

**focus** (ODTypeToken) - input

A tokenized string representing the focus type to be transferred.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus	The frame with the clipboard focus has access to the clipboard.
kODKeyFocus	The frame with keyboard focus receives keyboard events (excluding page-movement key events).
kODMenuFocus	The frame with menu focus receives menu events.
kODModalFocus	A frame with modal focus is notifying other frames that it is the only currently modal frame.
kODMouseFocus	The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.
kODScrollingFocus	The frame with the scrolling focus receives page-movement key events (such as <b>PageUp</b> ). This frame does not need to own keyboard focus.
kODSelectionFocus	The frame with the selection focus receives modified mouse-click events ( <b>Shift</b> +click and <b>Ctrl</b> +click). OpenDoc draws the active frame border around all facets of this frame.

---

## TransferFocusOwnership Parameter - transferringFrame

**transferringFrame** (ODFrame \*) - input

A reference to a frame relinquishing ownership of the focus; it does not have to be the current owner of the focus.

---

## TransferFocusOwnership Parameter - newOwner



**newOwner** (ODFrame \*) - input  
A reference to a frame obtaining ownership of the focus.

---

## TransferFocusOwnership - Return Value

None.

---

## TransferFocusOwnership - Parameters

**focus** (ODTypeToken) - input  
A tokenized string representing the focus type to be transferred.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

**kODClipboardFocus**  
The frame with the clipboard focus has access to the clipboard.

**kODKeyFocus**  
The frame with keyboard focus receives keyboard events (excluding page-movement key events).

**kODMenuFocus**  
The frame with menu focus receives menu events.

**kODModalFocus**  
A frame with modal focus is notifying other frames that it is the only currently modal frame.

**kODMouseFocus**  
The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

**kODScrollingFocus**  
The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

**kODSelectionFocus**  
The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

**transferringFrame** (ODFrame \*) - input  
A reference to a frame relinquishing ownership of the focus; it does not have to be the current owner of the focus.

**newOwner** (ODFrame \*) - input  
A reference to a frame obtaining ownership of the focus.

None.

---

## TransferFocusOwnership - Remarks

OpenDoc calls this method. This method should record, in the focus module's internal structures, the transfer of focus ownership.

---

## TransferFocusOwnership - Override Policy

If you subclass [ODFocusModule](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## TransferFocusOwnership - Related Methods

### Related Methods

- [ODSession::Tokenize](#)
- 

## TransferFocusOwnership - Topics

### Class:

[ODFocusModule](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Related Methods](#)

---

## UnsetFocusOwnership

---

## UnsetFocusOwnership - Syntax

This method should remove the specified frame as the owner of the specified focus.

```
#define INCL_ODFOCUSMODULE
#define INCL_ODAPI
#include <os2.h>

ODTypeToken    focus;
ODFrame        *frame;

UnsetFocusOwnership(focus, frame);
```

---

## UnsetFocusOwnership Parameter - focus

**focus** ([ODTypeToken](#)) - input

A tokenized string representing the focus whose owner is to be removed.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the clipboard focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus

The frame with menu focus receives menu events.

kODModalFocus

A frame with modal focus is notifying other frames that it is the only currently modal frame.

kODMouseFocus

The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

kODScrollingFocus

The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

kODSelectionFocus

The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

-----

## UnsetFocusOwnership Parameter - frame

**frame** (ODFrame \*) - input

A reference to the current owner of the focus.

-----

## UnsetFocusOwnership - Return Value

None.

-----

## UnsetFocusOwnership - Parameters

**focus** (ODTypeToken) - input

A tokenized string representing the focus whose owner is to be removed.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the clipboard focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus

The frame with menu focus receives menu events.

kODModalFocus

A frame with modal focus is notifying other frames that it is the only currently modal frame.

kODMouseFocus

The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

kODScrollingFocus

The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

kODSelectionFocus

The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

**frame** (ODFrame \*) - input

A reference to the current owner of the focus.

None.

-----

## UnsetFocusOwnership - Remarks

OpenDoc calls this method. This method should record, in the focus module's internal structures, the loss of focus ownership. This method can be called for both exclusive and nonexclusive foci.

-----

## UnsetFocusOwnership - Override Policy

If you subclass [ODFocusModule](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

-----

## UnsetFocusOwnership - Related Methods

### Related Methods

- [ODSession::Tokenize](#)

-----

## UnsetFocusOwnership - Topics

### Class:

[ODFocusModule](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Related Methods](#)

-----

## ODFocusOwnerIterator

**Class Definition File:** FOCUSOWN.IDL

### Class Hierarchy

SOMObject  
ODObject

## ODFocusOwnerIterator

### Description

An object of the ODFocusOwnerIterator class provides access to all owners of a nonexclusive focus.

Foci may be exclusive or nonexclusive. All of the standard foci defined by OpenDoc are *exclusive*, meaning that only one frame can own a focus at a time. If you create a new kind of focus, you can make it *nonexclusive*, meaning that several frames could share ownership of it.

You must create a focus-owner iterator if you create a focus module for a nonexclusive focus. The focus module keeps a list of all individual frames that own the nonexclusive focus. A focus-owner iterator contains a reference to the focus module that manages the nonexclusive focus and provides access to all of the owner frames.

You use a focus-owner iterator to apply an operation to all owners of a nonexclusive focus. For example, a part might use a focus-owner iterator to notify all frames that own a video input focus to synchronize themselves so that the video is displayed in all of the frames simultaneously.

The ODFocusOwnerIterator class is an abstract superclass that you can subclass to create a focus-owner iterator. Your part creates a focus-owner iterator object by calling the arbitrator's [CreateOwnerIterator](#) method. OpenDoc in turn calls the appropriate focus module's [CreateOwnerIterator](#) method, which returns a reference to a focus-owner iterator object.

While you are using a focus-owner iterator, you should not modify the list of focus owners. You must postpone adding frames to or removing frames from the list of focus owners until after you have deleted the iterator.

For more information on accessing objects through iterators, see the chapter on OpenDoc run-time features in the *OpenDoc Programming Guide*.

### Overriding Inherited Methods

The following methods are inherited and available for use by your subclass of ODFocusOwnerIterator.

#### somInit

The somInit method initializes the new instance variables in a SOM object; it is inherited from the SOMObject class.

If you subclass ODFocusOwnerIterator, you can override this method. Your override method does not need to call its inherited method; the inherited method is automatically called for you by the SOM library.

Your override of this method should initialize the instance variables in this focus-owner iterator object. The SOM library calls this method when this focus-owner iterator is created. You must not do anything that might cause this method to fail. This limits you to operations like setting pointer variables to null, setting numeric variables to appropriate values, and making similar assignments from constants. If you have any initialization code that can potentially fail, it must be handled in the focus-owner iterator's subclass-specific initialization method; see also the method [InitFocusOwnerIterator](#).

#### somUninit

The somUninit method disposes of the storage created for a SOM object; it is inherited from the SOMObject class.

If you subclass ODFocusOwnerIterator, you can override this method. Your override method does not need to call its inherited method; the inherited method is automatically called for you by the SOM library.

Your override method should dispose of any storage created for this focus-owner iterator object, including any storage related to additional instances variables initialized in this focus-owner iterator object. The SOM library calls this method when this focus-owner iterator is deleted; this method must not fail.

### Methods

The methods defined by the ODFocusOwnerIterator class include:

- [First](#)
- [InitFocusOwnerIterator](#)
- [IsNotComplete](#)
- [Next](#)

### Overridden Methods

There are currently no methods overridden by the ODFocusOwnerIterator class.

---

## First

---

## First - Syntax

This method should begin the iteration and return a reference to the first frame in the iteration sequence.

```
#define INCL_ODFOCUSOWNERITERATOR
#define INCL_ODAPI
#include <os2.h>

ODFrame      *rv;

rv = First();
```

---

## First Return Value - rv

**rv** (ODFrame \*) - returns

A reference to the first frame in the iteration sequence or kODDNULL if the focus has no owners.

---

## First - Parameters

**rv** (ODFrame \*) - returns

A reference to the first frame in the iteration sequence or kODDNULL if the focus has no owners.

---

## First - Remarks

Your part calls this method before calling this focus-owner iterator's [IsNotComplete](#) method for the first time. This method may be called multiple times; each time, it resets the iteration.

Your override of this method should not increment the reference count of the returned frame object.

---

## First - Exception Handling

kODErrIteratorOutOfSync

The list of focus owners was modified while the iteration was in progress.

---

## First - Override Policy

If you subclass [ODFocusOwnerIterator](#), you must override this method. Your override must not call its inherited method; that is, your override

must implement this method's functionality completely.

---

## First - Topics

### Class:

ODFocusOwnerIterator

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

---

## InitFocusOwnerIterator

---

### InitFocusOwnerIterator - Syntax

This method initializes this focus-owner iterator object.

```
#define INCL_ODFOCUSOWNERITERATOR
#define INCL_ODAPI
#include <os2.h>

ODTypeToken      focus;
ODFocusModule    *focusModule;

InitFocusOwnerIterator(focus, focusModule);
```

---

### InitFocusOwnerIterator Parameter - focus

**focus** ([ODTypeToken](#)) - input

A tokenized string representing the nonexclusive focus type of the owners returned by this iterator

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the [clipboard](#) focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus

The frame with menu focus receives menu events.

kODModalFocus

A frame with modal focus is notifying other frames that it is the only currently modal frame.

kODMouseFocus

The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

kODScrollingFocus

The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

kODSelectionFocus

The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

-----

## InitFocusOwnerIterator Parameter - focusModule

**focusModule** (ODFocusModule \*) - input

A reference to a focus module which lists the owners of the specified focus.

-----

## InitFocusOwnerIterator - Return Value

None.

-----

## InitFocusOwnerIterator - Parameters

**focus** (ODTypeToken) - input

A tokenized string representing the nonexclusive focus type of the owners returned by this iterator

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the clipboard focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus

The frame with menu focus receives menu events.

kODModalFocus

A frame with modal focus is notifying other frames that it is the only currently modal frame.

kODMouseFocus

The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

kODScrollingFocus

The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

kODSelectionFocus

The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.



**focusModule** (ODFocusModule \*) - input

A reference to a focus module which lists the owners of the specified focus.

None.

---

## InitFocusOwnerIterator - Remarks

This method is not called directly to initialize this focus-owner iterator object, but is called by a subclass-specific initialization method. By convention, every subclass of [ODFocusOwnerIterator](#) should have a separate initialization method (for example, the `InitMyFocusOwnerIterator` method) that is called when an instance of that subclass is created. The override method may have additional parameters beyond those of the `InitFocusOwnerIterator` method. The `InitMyFocusOwnerIterator` method should call the inherited `InitFocusOwnerIterator` method at the beginning of its implementation.

If you subclass [ODFocusOwnerIterator](#), your subclass-specific initialization method, rather than its `somInit` method, should handle any initialization code that can potentially fail. For example, your initialization method may attempt to allocate memory for your focus-owner iterator.

---

## InitFocusOwnerIterator - Override Policy

If you subclass [ODFocusOwnerIterator](#), you must not override this method.

---

## InitFocusOwnerIterator - Related Methods

### Related Methods

- [ODSession::Tokenize](#)
- 

## InitFocusOwnerIterator - Topics

### Class:

[ODFocusOwnerIterator](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Related Methods](#)

---

## IsNotComplete

---

# IsNotComplete - Syntax

This method should indicate whether the iteration is incomplete.

```
#define INCL_ODFOCUSOWNERITERATOR
#define INCL_ODAPI
#include <os2.h>

ODBoolean    rv;

rv = IsNotComplete();
```

---

## IsNotComplete Return Value - rv

**rv** ([ODBoolean](#)) - returns  
A flag indicating whether the iteration is incomplete.

kODTrue	The iteration is incomplete.
kODFalse	The iteration is complete.

---

## IsNotComplete - Parameters

**rv** ([ODBoolean](#)) - returns  
A flag indicating whether the iteration is incomplete.

kODTrue	The iteration is incomplete.
kODFalse	The iteration is complete.

---

## IsNotComplete - Remarks

Your part calls this method to test whether more focus owners remain in the iteration sequence. This method returns kODTrue if the preceding call to the [First](#) or [Next](#) method found a focus owner. This method returns kODFalse when you have examined all the focus owners (that is, when the previous call to [First](#) or [Next](#) returned kODNULL).

---

## IsNotComplete - Exception Handling

kODErrIteratorNotInitialized

This method was called before calling either the [First](#) or [Next](#) method to begin the iteration.

kODErrIteratorOutOfSync

The list of focus owners was modified while the iteration was in progress.

---

## IsNotComplete - Override Policy

If you subclass [ODFocusOwnerIterator](#), you must override this method. Your override must not call its inherited method; that is, your override must implement this method's functionality completely.

---

## IsNotComplete - Topics

### Class:

[ODFocusOwnerIterator](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

---

## Next

---

## Next - Syntax

This method should return a reference to the next frame in the iteration sequence.

```
#define INCL_ODFOCUSOWNERITERATOR
#define INCL_ODAPI
#include <os2.h>
```

```
ODFrame      *rv;
```

```
rv = Next();
```

---

## Next Return Value - rv

**rv** (ODFrame \*) - returns

A reference to the next frame in the iteration sequence or KODNULL if you have reached the last frame.

---

## Next - Parameters

**rv** (ODFrame \*) - returns

A reference to the next frame in the iteration sequence or kODNULL if you have reached the last frame.

-----

## Next - Remarks

Your part calls this method. If your part calls this method before calling this focus-owner iterator's [First](#) method to begin the iteration, this method works the same as calling the [First](#) method.

Your override of this method should not increment the reference count of the returned frame object.

-----

## Next - Exception Handling

kODErrIteratorOutOfSync

The list of focus owners was modified while the iteration was in progress.

-----

## Next - Override Policy

If you subclass [ODFocusOwnerIterator](#), you must override this method. Your override must not call its inherited method; that is, your override must implement this method's functionality completely.

-----

## Next - Topics

### Class:

ODFocusOwnerIterator

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

-----

## ODFocusSet

**Class Definition File:** FOCUSSET.IDL

### Class Hierarchy

SOMObject

ODObject

**ODFocusSet**

### Description

An object of the `ODFocusSet` class provides grouping of foci for activation.

A *focus* is a designation of ownership of a given shared resource or feature, such as the keyboard or menu bar. A frame that owns a focus receives events pertaining to that resource. Foci can be manipulated individually or in groups called focus sets. A *focus set* is a list of foci that can be obtained or released as a group. For example, if a frame wants to request ownership of keyboard and menu foci together, it can create a focus set that includes these two focus types. If a frame requests a focus set and one or more of the foci included in the focus set is not available, ownership of the entire focus set is denied.

Your part creates a focus set object by calling the arbitrator's `CreateFocusSet` method, which returns a reference to a focus set object. Frames obtain ownership of focus sets by calling the arbitrator's `RequestFocusSet` method.

For more information about foci and focus sets, see the class description for `ODFocusSetIterator` and the chapter on user events in the *OpenDoc Programming Guide*.

## Methods

The methods defined by the `ODFocusSet` class include:

- [Add](#)
- [Contains](#)
- [CreateIterator](#)
- [Remove](#)

## Overridden Methods

There are currently no methods overridden by the `ODFocusSet` class.

---

# Add

---

## Add - Syntax

This method adds the specified focus to this focus set.

```
#define INCL_ODFOCUSSET
#define INCL_ODAPI
#include <os2.h>

ODTypeToken    focus;

Add(focus);
```

---

## Add Parameter - focus

**focus** ([ODTypeToken](#)) - input

A tokenized string representing a focus type to be added to this focus set.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

`kODClipboardFocus`

The frame with the clipboard focus has access to the clipboard.

`kODKeyFocus`

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

`kODMenuFocus`

The frame with menu focus receives menu events.

kODModalFocus	A frame with modal focus is notifying other frames that it is the only currently modal frame.
kODMouseFocus	The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.
kODScrollingFocus	The frame with the scrolling focus receives page-movement key events (such as <b>PageUp</b> ). This frame does not need to own keyboard focus.
kODSelectionFocus	The frame with the selection focus receives modified mouse-click events ( <b>Shift</b> +click and <b>Ctrl</b> +click). OpenDoc draws the active frame border around all facets of this frame.

-----

## Add - Return Value

None.

-----

## Add - Parameters

**focus** (ODTypeToken) - input

A tokenized string representing a focus type to be added to this focus set.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus	The frame with the clipboard focus has access to the clipboard.
kODKeyFocus	The frame with keyboard focus receives keyboard events (excluding page-movement key events).
kODMenuFocus	The frame with menu focus receives menu events.
kODModalFocus	A frame with modal focus is notifying other frames that it is the only currently modal frame.
kODMouseFocus	The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.
kODScrollingFocus	The frame with the scrolling focus receives page-movement key events (such as <b>PageUp</b> ). This frame does not need to own keyboard focus.
kODSelectionFocus	The frame with the selection focus receives modified mouse-click events ( <b>Shift</b> +click and <b>Ctrl</b> +click). OpenDoc draws the active frame border around all facets of this frame.

None.

-----

## Add - Remarks

The specified focus is not added if it already exists in the focus set.

-----

## Add - Exception Handling

kODErrOutOfMemory

There is not enough memory to expand the focus set.

---

## Add - Related Methods

### Related Methods

- [ODSession::Tokenize](#)

---

## Add - Topics

### Class:

ODFocusSet

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

## Contains

---

## Contains - Syntax

This method indicates whether the specified focus is a member of this focus set.

```
#define INCL_ODFOCUSSET
#define INCL_ODAPI
#include <os2.h>
```

```
ODTypeToken    focus;
ODBoolean      rv;
```

```
rv = Contains(focus);
```

---

## Contains Parameter - focus

**focus** (ODTypeToken) - input

A tokenized string representing a focus type to be tested for membership.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the clipboard focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus

The frame with menu focus receives menu events.

kODModalFocus

A frame with modal focus is notifying other frames that it is the only currently modal frame.

kODMouseFocus

The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

kODScrollingFocus

The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

kODSelectionFocus

The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

-----

## Contains Return Value - rv

**rv** (ODBoolean) - returns

A flag indicating whether the specified focus is a member of this focus set.

kODTrue

The set contains the focus.

kODFalse

The set does not contain the focus.

-----

## Contains - Parameters

**focus** (ODTypeToken) - input

A tokenized string representing a focus type to be tested for membership.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the clipboard focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus

The frame with menu focus receives menu events.

kODModalFocus

A frame with modal focus is notifying other frames that it is the only currently modal frame.

kODMouseFocus

The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

kODScrollingFocus

The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

kODSelectionFocus

The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

**rv** (ODBoolean) - returns

A flag indicating whether the specified focus is a member of this focus set.



kODTrue	The set contains the focus.
kODFalse	The set does not contain the focus.

---

## Contains - Related Methods

### Related Methods

- [ODSession::Tokenize](#)
- 

## Contains - Topics

### Class:

[ODFocusSet](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Related Methods](#)

---

## CreateIterator

---

## CreateIterator - Syntax

This method creates a focus-set iterator object for this focus set.

```
#define INCL_ODFOCUSSET
#define INCL_ODAPI
#include <os2.h>

ODFocusSetIterator      *rv;

rv = CreateIterator();
```

---

## CreateIterator Return Value - rv

**rv** (ODFocusSetIterator \*) - returns  
 A reference to a new focus-set iterator object.

---

## CreateIterator - Parameters

**rv** (ODFocusSetIterator \*) - returns  
A reference to a new focus-set iterator object.

---

## CreateIterator - Remarks

While you are using a focus-set iterator, you should not modify the focus set. You must postpone adding items to or removing items from the focus set until after you have deleted the iterator.

---

## CreateIterator - Topics

**Class:**  
ODFocusSet

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

## Remove

---

## Remove - Syntax

This method removes the specified focus from this focus set.

```
#define INCL_ODFOCUSSET
#define INCL_ODAPI
#include <os2.h>
```

```
ODTypeToken    focus;
```

```
Remove(focus);
```

---

## Remove Parameter - focus

**focus** (ODTypeToken) - input

A tokenized string representing the focus type to be remove from the focus set.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the clipboard focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus

The frame with menu focus receives menu events.

kODModalFocus

A frame with modal focus is notifying other frames that it is the only currently modal frame.

kODMouseFocus

The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

kODScrollingFocus

The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

kODSelectionFocus

The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

-----

## Remove - Return Value

None.

-----

## Remove - Parameters

**focus** (ODTypeToken) - input

A tokenized string representing the focus type to be remove from the focus set.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the clipboard focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus

The frame with menu focus receives menu events.

kODModalFocus

A frame with modal focus is notifying other frames that it is the only currently modal frame.

kODMouseFocus

The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

kODScrollingFocus

The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

kODSelectionFocus

The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

None.

---

# Remove - Related Methods

## Related Methods

- [ODSession::Tokenize](#)
- 

# Remove - Topics

## Class:

ODFocusSet

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Related Methods](#)

---

# ODFocusSetIterator

**Class Definition File:** FOCUSITR.IDL

## Class Hierarchy

SOMObject  
ODObject  
    **ODFocusSetIterator**

## Description

An object of the ODFocusSetIterator class provides access to all foci in a focus set.

You use a focus-set iterator to apply an operation to all foci in a focus set. For example, you might store a focus set with all the foci owned by a given frame and then use a focus-set iterator to search for a particular focus in a focus set.

Your part creates a focus-set iterator object by calling a focus set's [CreateIterator](#) method, which returns a reference to a focus-set iterator object.

While you are using a focus-set iterator, you should not modify or delete the focus set. You must postpone adding foci to or removing foci from the focus set until after you have delete the iterator.

For more information on accessing objects through iterators, see the chapter on OpenDoc run-time features in the *OpenDoc Programming Guide* .

## Methods

The methods defined by the ODFocusSetIterator class include:

- [First](#)
- [IsNotComplete](#)
- [Next](#)

## Overridden Methods

There are currently no methods overridden by the ODFocusSetIterator class.

---

# First

---

## First - Syntax

This method begins the iteration and returns the first focus in the focus set.

```
#define INCL_ODFOCUSSET
#define INCL_ODAPI
#include <os2.h>

ODTypeToken    rv;

rv = First();
```

---

## First Return Value - rv

**rv** ([ODTypeToken](#)) - returns

A tokenized string representing the first focus in the focus set or kODNullFocus if the focus set is empty.

---

## First - Parameters

**rv** ([ODTypeToken](#)) - returns

A tokenized string representing the first focus in the focus set or kODNullFocus if the focus set is empty.

---

## First - Remarks

Your part must call this method before calling this focus-set iterator's [IsNotComplete](#) method for the first time. This method may be called multiple times; each time, it resets the iteration.

---

## First - Exception Handling

kODErrIteratorOutOfSync

The focus set was modified while the iteration was in progress.

---

## First - Topics

**Class:**  
ODFocusSetIterator

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

---

## IsNotComplete

---

## IsNotComplete - Syntax

This method indicates whether the iteration is incomplete.

```
#define INCL_ODFOCUSSET
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    rv;
```

```
rv = IsNotComplete();
```

---

## IsNotComplete Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the iteration is incomplete.

kODTrue

The iteration is incomplete.

kODFalse

The iteration is complete.

---

## IsNotComplete - Parameters

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the iteration is incomplete.

kODTrue

The iteration is incomplete.

kODFalse

The iteration is complete.

---

# IsNotComplete - Remarks

Your part calls this method to test whether more foci remain in the focus set. This method returns kODTrue if the preceding call to the [First](#) or [Next](#) method found a focus. This method returns kODFalse when you have examined all the foci (that is, when the previous call to [First](#) or [Next](#) returned kODNULL). If the focus set is empty, this method always returns kODFalse.

-----

# IsNotComplete - Exception Handling

kODErrIteratorNotInitialized

This method was called before calling either the [First](#) or [Next](#) method to begin the iteration.

kODErrIteratorOutOfSync

The focus set was modified while the iteration was in progress.

-----

# IsNotComplete - Topics

## Class:

ODFocusSetIterator

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

-----

# Next

-----

# Next - Syntax

This method returns a reference to the next focus in the focus set.

```
#define INCL_ODFOCUSSET
#define INCL_ODAPI
#include <os2.h>
```

```
ODTypeToken    rv;
```

```
rv = Next();
```

-----

# Next Return Value - rv

**rv** ([ODTypeToken](#)) - returns

A tokenized string representing the next focus in the focus set or kODNullFocus if you have reached the end of the focus set.

---

## Next - Parameters

**rv** ([ODTypeToken](#)) - returns

A tokenized string representing the next focus in the focus set or kODNullFocus if you have reached the end of the focus set.

---

## Next - Remarks

If your part calls this method before calling this focus-set iterator's [First](#) method to begin the iteration, this method works the same as calling the [First](#) method.

---

## Next - Exception Handling

kODErrIteratorOutOfSync

The focus set was modified while the iteration was in progress.

---

## Next - Topics

### Class:

ODFocusSetIterator

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

---

## ODFrame

**Class definition file:** FRAME.IDL

### Class hierarchy

SOMObject

ODObject

ODRefCntObject

ODPersistentObject

**ODFrame**

### Description



An object of the `ODFrame` class describes the display area of an embedded part in the content of its containing part.

A frame object represents an area of content of a part; it marks the geometric boundary between an embedded part and its containing part. There are several possible configurations and states for frames:

Active frame	A frame that has the selection focus. Editing takes place in the active frame; that frame displays the selection or insertion point.
Bundled frame	A frame whose content does not respond to geometry-based user events. A mouse click within a bundled frame selects the frame's part, but does not activate it.
Containing frame	A frame in which one or more frames are embedded. Each embedded frame has one containing frame; each containing frame has one or more embedded frames.
Display frame	A frame within which the part's content is drawn. Each display frame represents a particular view of a part's content.
Embedded frame	A frame within which one of the part's embedded parts is displayed.
Nonpersistent frame	A frame that exists only as an in-memory object. A nonpersistent frame does not have a storage unit and is not stored persistently.
Overlaid frame	An embedded frame that floats above a containing part's content (including other embedded frames). Overlaid frames do not need to negotiate for space except as required by the constraints of the containing part size.
Root frame	A frame in which the root part of a window is displayed. The root frame shape is the same as the content area of the window.
Subframe	A frame that is both an embedded frame in, and a display frame of, a part. A part can create an embedded frame, make it a subframe of its own display frame, and then display itself in that subframe. When a subframe has the selection focus, the active frame border is displayed around its containing frame.

Your part creates a new frame object for its embedded parts by calling its draft's [CreateFrame](#) method. Your part accesses a previously stored frame object by calling its draft's [AcquireFrame](#) method. These methods return a reference to a frame object.

## Frame Geometry

There are several things that define the geometry of a frame.

Content extent	This describes how much offset is used to calculate the bias transform—that is, the vertical extent of the content area of a part in a frame (in essence, the height of a part's page). For more information related to bias transforms, see the class description of the <a href="#">ODCanvas</a> .
Frame shape	This defines the area that the containing part propagates to an embedded part's display frame. The frame's containing part controls the frame shape.
Internal transform	This describes how the part's content is positioned, scaled, or otherwise transformed within the frame. It represents the mapping from the coordinate space of the frame's content to the coordinate space of the frame.
Used shape	This defines the area of an embedded part's frame that has actual content to display—that is, the part of the frame that the containing part should not draw over; however, the containing part is free to wrap content to the contour of the used shape. The embedded part controls the used shape.

A frame must always possess a valid frame shape. The used shape does not need to be set; if it is not, the used shape is the same as the frame shape. For more information related to shape objects, see the class description of [ODShape](#).

A frame must always possess a valid internal transform which, if not explicitly set, is the identity transform. For more information related to transform objects, see the class description of [ODTransform](#).

The `ODFrame` class includes several methods that specify geometry (shape and transform) objects. Because these objects necessarily assume a coordinate system, the `ODFrame` methods include a *biasCanvas* parameter that allows you to specify a canvas to whose coordinate space the geometry is biased. The *bias canvas* uses a bias transform to convert from the coordinate system used for drawing on the canvas to the coordinate system (platform-normal coordinates) used by the current graphics system. When the bias canvas is specified, it is automatically applied to the returned object.

Each part in an OpenDoc document controls the position, size, and shape of the frames embedded within it. At the same time, embedded parts may want to change the size, shape, or number of frames they are displayed in. Through a process called *frame negotiation*, an embedded part and its containing part agree on the frames the part displays in. Either part can initiate the negotiation, although the containing part has unilateral control over the outcome.

## Frame Hierarchy

A frame always maintains a reference to its part. The frame ensures that it is registered with its part on that part's internal list of display frames when the frame is created (by calling its part's [DisplayFrameAdded](#) method) and that it is removed from that part's internal list of display frames when the frame is deleted (by calling its part's [DisplayFrameRemoved](#) method).

A frame maintains a reference to its containing frame. The value of the reference is null if the specified frame is the root frame of a window. The reference value does not usually change, unless the frame was created before an embedding location was selected. A frame does not hold direct references to its embedded frames. Only the frame's part knows which frames are embedded in the frame.

A frame maintains a list of its facets. Facets hold nonpersistent information about the layout of parts, or describe the location of a frame on a particular canvas for display and event dispatching. There may be more than one facet per frame. This list may be empty if the specified frame is currently scrolled out of view or otherwise not visible. For more information related to facet objects, see the class description for [ODFacet](#).

## Displaying Information

There are two characteristics that describe how a part is displayed within a frame.

- The *view type* describes the basic visual representation of a part. Parts must support the standard set of view types (large icon, small icon, thumbnail, or frame view). The view type should be set only by the frame's part.
- The *presentation* of a frame describes, for parts whose view type is framed, a particular style of display for a part's content within the frame—for example, a table, bar chart, pie chart, text, or outline. Presentations are part-defined; your part editor determines what types of presentations your part is capable of and defines a presentation designation for each. A containing part may request a particular presentation but the embedded part does not need to honor that request. The presentation should be set only by the frame's part.

A frame's part might store user-defined data in the frame's part information. The part alone interprets or manipulates the data, but the part-information data is stored with the frame and read in only when it is necessary for frame manipulation.

## Propagating Events

If one of your part's embedded frames propagates unhandled events to your part, your part has the opportunity to handle those events. In that case, your part's event handler needs to determine if a particular frame is one of its display frames or an embedded frame.

## Frame Groups

A *frame group* is a set of display frames that a part designates as related, for purposes such as flowing content from one frame to another. Each frame group has its own *group ID*; frames within a frame group have a *sequence number* that is used to define the position of a frame within its frame group. The group ID and sequence number should be set only by the frame's containing part.

## Flags and Link Status

When a part creates a frame, the part sets flags, defined for the lifetime of the frame, that determine whether the frame is a root frame, a subframe, or an overlaid frame; once set, these flags cannot be changed.

A containing part must set the link status for any frame it embeds, even if the embedded part does not support links to its content. The containing part specifies a link status value based solely on the embedded frame's inclusion in links maintained by the containing part. A frame considers the link status of its containing frame when setting its own link status.

## New methods

The methods defined for the `ODFrame` class include:

- [AcquireContainingFrame](#)
- [AcquireFrameShape](#)
- [AcquireInternalTransform](#)
- [AcquirePart](#)
- [AcquireUsedShape](#)
- [AcquireWindow](#)
- [ChangeContentExtent](#)
- [ChangeFrameShape](#)
- [ChangeInternalTransform](#)
- [ChangeLinkStatus](#)
- [ChangePart](#)
- [ChangePresentation](#)
- [ChangeSequenceNumber](#)
- [ChangeUsedShape](#)
- [ChangeViewType](#)
- [Close](#)
- [ContentUpdated](#)
- [CreateFacetIterator](#)
- [CreateShape](#)
- [CreateTransform](#)
- [DoesPropagateEvents](#)
- [DrawActiveBorder](#)
- [EditInLink](#)

- [FacetAdded](#)
- [FacetRemoved](#)
- [GetContentExtent](#)
- [GetFrameGroup](#)
- [GetLinkStatus](#)
- [GetPartInfo](#)
- [GetPresentation](#)
- [GetSequenceNumber](#)
- [GetViewType](#)
- [Invalidate](#)
- [InvalidateActiveBorder](#)
- [IsDragging](#)
- [IsDroppable](#)
- [IsFrozen](#)
- [IsInLimbo](#)
- [IsOverlaid](#)
- [IsRoot](#)
- [IsSubframe](#)
- [Remove](#)
- [RequestFrameShape](#)
- [SetContainingFrame](#)
- [SetDragging](#)
- [SetDroppable](#)
- [SetFrameGroup](#)
- [SetFrozen](#)
- [SetInLimbo](#)
- [SetPartInfo](#)
- [SetPresentation](#)
- [SetPropagateEvents](#)
- [SetSubframe](#)
- [SetViewType](#)
- [SetWindow](#)
- [Validate](#)

#### Overridden methods

There are currently no methods overridden by the ODFrame class.

---

## AcquireContainingFrame

---

## AcquireContainingFrame - Syntax

This method returns a reference to the containing frame of this frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODFrame      *rv;

rv = AcquireContainingFrame();
```

---

## AcquireContainingFrame Return Value - rv

**rv** (ODFrame \*) - returns

A reference to the containing frame of this frame or kODNULL if this frame is a root frame.

---

## AcquireContainingFrame - Parameters

**rv** (ODFrame \*) - returns

A reference to the containing frame of this frame or kODNULL if this frame is a root frame.

---

## AcquireContainingFrame - Remarks

This method increments the reference count of the returned frame. When you have finished using that frame object, you should call its [Release](#) method.

---

## AcquireContainingFrame - Related Methods

### Related Methods

- [ODFrame::SetContainingFrame](#)
- 

## AcquireContainingFrame - Topics

### Class:

ODFrame

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## AcquireFrameShape

---

## AcquireFrameShape - Syntax

This method returns a reference to the frame shape of this frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODCanvas      *biasCanvas;
ODShape       *rv;

rv = AcquireFrameShape(biasCanvas);
```

---

## AcquireFrameShape Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates.

---

## AcquireFrameShape Return Value - rv

**rv** (ODShape \*) - returns

A reference to the frame shape, expressed in frame coordinates.

---

## AcquireFrameShape - Parameters

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates.

**rv** (ODShape \*) - returns

A reference to the frame shape, expressed in frame coordinates.

---

## AcquireFrameShape - Remarks

The caller must not modify the returned shape. Only the frame's containing part can modify the frame shape.

This method increments the reference count of the returned shape object. When you have finished using that shape object, you should call its [Release](#) method.

---

## AcquireFrameShape - Related Methods

### Related Methods

- [ODFrame::ChangeFrameShape](#)
- [ODFrame::RequestFrameShape](#)

---

## AcquireFrameShape - Topics

### Class:

ODFrame

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

## AcquireInternalTransform

---

## AcquireInternalTransform - Syntax

This method returns a reference to an internal transform of this frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODCanvas      *biasCanvas;
ODTransform    *rv;

rv = AcquireInternalTransform(biasCanvas);
```

---

## AcquireInternalTransform Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates.

---

## AcquireInternalTransform Return Value - rv

**rv** (ODTransform \*) - returns

A reference to the internal transform of this frame.

---

# AcquireInternalTransform - Parameters

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in platform-normal coordinates.

**rv** (ODTransform \*) - returns

A reference to the internal transform of this frame.

---

## AcquireInternalTransform - Remarks

The caller must not modify the internal transform. Only this frame's part can modify the internal transform.

This method increments the reference count of the returned transform object. When you have finished using that transform object, you should call its [Release](#) method.

---

## AcquireInternalTransform - Related Methods

### Related Methods

- [ODFrame::ChangeInternalTransform](#)

---

## AcquireInternalTransform - Topics

### Class:

ODFrame

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## AcquirePart

---

## AcquirePart - Syntax

This method returns a reference to a part displayed in this frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODPart      *rv;

rv = AcquirePart();
```

---

## AcquirePart Return Value - rv

**rv** (ODPart \*) - returns  
A reference to a part displayed in this frame.

---

## AcquirePart - Parameters

**rv** (ODPart \*) - returns  
A reference to a part displayed in this frame.

---

## AcquirePart - Remarks

This method increments the reference count of the returned part object. When you have finished using that part object, you should call its [Release](#) method.

---

## AcquirePart - Related Methods

### Related Methods

- [ODFrame::ChangePart](#)

---

## AcquirePart - Topics

**Class:**  
ODFrame

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)



---

# AcquireUsedShape

---

## AcquireUsedShape - Syntax

This method returns a reference to the used shape of this frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODCanvas      *biasCanvas;
ODShape       *rv;

rv = AcquireUsedShape(biasCanvas);
```

---

## AcquireUsedShape Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates.

---

## AcquireUsedShape Return Value - rv

**rv** (ODShape \*) - returns

A reference to the used shape of this frame, expressed in frame coordinates.

---

## AcquireUsedShape - Parameters

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates.

**rv** (ODShape \*) - returns

A reference to the used shape of this frame, expressed in frame coordinates.

---

## AcquireUsedShape - Remarks

The caller must not modify the used shape. Only the frame's part can modify the used shape.

This method increments the reference count of the returned shape object. When you have finished using that shape object, you should call its [Release](#) method.

---

## AcquireUsedShape - Related Methods

### Related Methods

- [ODFrame::ChangeUsedShape](#)

---

## AcquireUsedShape - Topics

### Class:

ODFrame

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## AcquireWindow

---

## AcquireWindow - Syntax

This method returns a reference to the window in which this frame is displayed.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>
```

```
ODWindow      *rv;
```

```
rv = AcquireWindow();
```

---

## AcquireWindow Return Value - rv

**rv** (ODWindow \*) - returns

A reference to the window in which this frame is displayed or KODNULL if this frame does not actually appear in any window (for example, a printing frame does not appear in a window).

---

## AcquireWindow - Parameters

**rv** (ODWindow \*) - returns

A reference to the window in which this frame is displayed or KODNULL if this frame does not actually appear in any window (for example, a printing frame does not appear in a window).

---

## AcquireWindow - Remarks

Only the root frame of a window has a reference to the window; all its embedded frames then inherit this value.

This method increments the reference count of the returned window object. When you have finished using that window object, you should call its [Release](#) method.

---

## AcquireWindow - Related Methods

### Related Methods

- [ODFacet::GetWindow](#)
  - [ODFrame::SetWindow](#)
- 

## AcquireWindow - Topics

### Class:

ODFrame

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## ChangeContentExtent

---

## ChangeContentExtent - Syntax

This method changes the content extent of this frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODPoint      *contentExtent;

ChangeContentExtent (contentExtent);
```

---

## ChangeContentExtent Parameter - contentExtent

**contentExtent** ([ODPoint \\*](#)) - input  
The content extent to be assigned to this frame.

---

## ChangeContentExtent - Return Value

None.

---

## ChangeContentExtent - Parameters

**contentExtent** ([ODPoint \\*](#)) - input  
The content extent to be assigned to this frame.

None.

---

## ChangeContentExtent - Remarks

The content extent is, in essence, the page height of the part displayed in this frame. This method causes the bias transform associated with this frame's content to be recomputed.

---

## ChangeContentExtent - Related Methods

### Related Methods

- [ODFrame::GetContentExtent](#)

---

## ChangeContentExtent - Topics

### Class:

ODFrame

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

## ChangeFrameShape

---

## ChangeFrameShape - Syntax

This method changes the frame shape of this frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODShape      *shape;
ODCanvas     *biasCanvas;

ChangeFrameShape(shape, biasCanvas);
```

---

## ChangeFrameShape Parameter - shape

**shape** (ODShape \*) - input

A reference to a frame shape to be assigned to this frame, expressed in frame coordinates.

---

## ChangeFrameShape Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates.

---

## ChangeFrameShape - Return Value

None.

---

## ChangeFrameShape - Parameters

**shape** (ODShape \*) - input

A reference to a frame shape to be assigned to this frame, expressed in frame coordinates.

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in platform-normal coordinates.

None.

---

## ChangeFrameShape - Remarks

Your part calls its embedded frame's ChangeFrameShape method when it changes the shape of the embedded frame. This method in turn calls the [FrameShapeChanged](#) method of the embedded frame's part to notify the part that its frame shape has changed.

If the used shape is the same as the frame shape, this method also calls the [UsedShapeChanged](#) method of the embedded frame's part to notify the part that its used shape has changed.

---

## ChangeFrameShape - Exception Handling

kODErrIllegalNullShapeInput

The *shape* parameter is null.

---

## ChangeFrameShape - Related Methods

### Related Methods

- [ODFrame::AcquireFrameShape](#)
  - [ODFrame::RequestFrameShape](#)
  - [ODPart::FrameShapeChanged](#)
  - [ODPart::UsedShapeChanged](#)
- 

## ChangeFrameShape - Topics

### Class:

ODFrame

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## ChangeInternalTransform

---

### ChangeInternalTransform - Syntax

This method changes the internal transform of this frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODTransform      *transform;
ODCanvas         *biasCanvas;

ChangeInternalTransform(transform, biasCanvas);
```

---

### ChangeInternalTransform Parameter - transform

**transform** (ODTransform \*) - input  
A reference to a internal transform to be associated with this frame.

---

### ChangeInternalTransform Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input  
A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates.

---

### ChangeInternalTransform - Return Value

None.

---

## ChangeInternalTransform - Parameters

**transform** (ODTransform \*) - input

A reference to a internal transform to be associated with this frame.

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in platform-normal coordinates.

None.

---

## ChangeInternalTransform - Remarks

Your part calls its display frame's ChangeInternalTransform method to change the position of its content in the frame.

---

## ChangeInternalTransform - Exception Handling

kODerrIllegalNullTransformInput

The *transform* parameter is null.

---

## ChangeInternalTransform - Related Methods

### Related Methods

- [ODFrame::AcquireInternalTransform](#)
- 

## ChangeInternalTransform - Topics

### Class:

ODFrame

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

## ChangeLinkStatus



---

## ChangeLinkStatus - Syntax

This method changes the link status of this frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODLinkStatus    status;

ChangeLinkStatus(status);
```

---

## ChangeLinkStatus Parameter - status

**status** ([ODLinkStatus](#)) - input

The link status to be assigned to this frame. This parameter can be set to one of the following values:

kODInLinkDestination

The frame is embedded in the destination of a link; the content of this frame is thus supplied by a link.

kODInLinkSource

The frame is embedded in the content that is the source of one or more links, but not in the content that is the destination of a link.

kODNotInLink

The frame is not embedded in any link content, source or destination.

---

## ChangeLinkStatus - Return Value

None.

---

## ChangeLinkStatus - Parameters

**status** ([ODLinkStatus](#)) - input

The link status to be assigned to this frame. This parameter can be set to one of the following values:

kODInLinkDestination

The frame is embedded in the destination of a link; the content of this frame is thus supplied by a link.

kODInLinkSource

The frame is embedded in the content that is the source of one or more links, but not in the content that is the destination of a link.

kODNotInLink

The frame is not embedded in any link content, source or destination.

None.

-----

## ChangeLinkStatus - Remarks

If your part supports linking or embedding, your part calls this method for each embedded frame that is involved in a link when a link is created, broken or moved. If your part supports linking, your part calls this method for each frame not in a link. This method in turn calls the [LinkStatusChanged](#) method associated with this frame's part to notify the part that its link status has changed. In turn, the embedded part's [LinkStatusChanged](#) method gives that embedded part a chance to call the [LinkStatusChanged](#) method for other embedded frames.

-----

## ChangeLinkStatus - Exception Handling

kODErrNotInLink

The specified link status is invalid.

-----

## ChangeLinkStatus - Related Methods

### Related Methods

- [ODFrame::GetLinkStatus](#)
- [ODPart::LinkStatusChanged](#)

-----

## ChangeLinkStatus - Topics

### Class:

ODFrame

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

-----

## ChangePart

-----

## ChangePart - Syntax

This method assigns a new part to this frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODPart      *part;

ChangePart (part);
```

---

## ChangePart Parameter - part

**part** (ODPart \*) - input  
A reference to a part to be associated with this frame.

---

## ChangePart - Return Value

None.

---

## ChangePart - Parameters

**part** (ODPart \*) - input  
A reference to a part to be associated with this frame.

None.

---

## ChangePart - Remarks

Your part calls this method when it wants to keep one embedded frame and swap parts in and out of it, as is typical for browser-type parts. This method in turn calls the [FacetRemoved](#) method associated with this frame's part to remove all facets of this frame from its previous part. This method can also call the [FacetAdded](#) method associated with this frame's new part to add all facets associated with the previous part.

Unless this frame was removed or closed, you can always assume that this frame previously had a part.

---

## ChangePart - Exception Handling

kODErrIllegalNullPartInput

The *part* parameter is null.

---

## ChangePart - Related Methods

### Related Methods

- [ODPart::FacetAdded](#)
  - [ODPart::FacetRemoved](#)
- 

## ChangePart - Topics

### Class:

[ODFrame](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## ChangePresentation

---

## ChangePresentation - Syntax

This method changes the presentation of this frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODTypeToken    presentation;

ChangePresentation(presentation);
```

---

## ChangePresentation Parameter - presentation

**presentation** ([ODTypeToken](#)) - input

A tokenized string representing the presentation to be assigned to this frame. The presentation values are determined by the part developer. They must be tokenized to ensure they are unique. Non-unique or null values are invalid.

---

## ChangePresentation - Return Value

None.

---

## ChangePresentation - Parameters

**presentation** ([ODTypeToken](#)) - input

A tokenized string representing the presentation to be assigned to this frame. The presentation values are determined by the part developer. They must be tokenized to ensure they are unique. Non-unique or null values are invalid.

None.

---

## ChangePresentation - Remarks

Your part calls its embedded frame's `ChangePresentation` method to request that it use a new presentation. This method in turn calls the [PresentationChanged](#) method of this frame's part to notify the part that its presentation has changed. If the embedded part does not support the requested presentation, it can pick a presentation that it can support and then call its frame's [SetPresentation](#) method to update the presentation in the frame.

---

## ChangePresentation - Exception Handling

`kODErrIllegalNullTokenInput`

The *presentation* parameter is null.

---

## ChangePresentation - Related Methods

### Related Methods

- [ODFrame::GetPresentation](#)
  - [ODFrame::SetPresentation](#)
  - [ODPart::PresentationChanged](#)
- 

## ChangePresentation - Topics

**Class:**  
ODFrame

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

---

# ChangeSequenceNumber

---

## ChangeSequenceNumber - Syntax

This method assigns a sequence number to this frame in its frame group.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODULong    sequenceNumber;

ChangeSequenceNumber (sequenceNumber);
```

---

## ChangeSequenceNumber Parameter - sequenceNumber

**sequenceNumber** ([ODULong](#)) - input  
The sequence number to be assigned to this frame.

---

## ChangeSequenceNumber - Return Value

None.

---

## ChangeSequenceNumber - Parameters

**sequenceNumber** ([ODULong](#)) - input  
The sequence number to be assigned to this frame.

None.

---

## ChangeSequenceNumber - Remarks

Your part calls its embedded frame's ChangeSequenceNumber method to reorder the sequence of its display frames in its frame group.

---

## ChangeSequenceNumber - Related Methods

### Related Methods

- [ODFrame::GetSequenceNumber](#)
- 

## ChangeSequenceNumber - Topics

### Class:

ODFrame

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## ChangeUsedShape

---

## ChangeUsedShape - Syntax

This method assigns a used shape to this frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODShape      *shape;
ODCanvas     *biasCanvas;

ChangeUsedShape(shape, biasCanvas);
```

---

## ChangeUsedShape Parameter - shape

**shape** (ODShape \*) - input

A reference to a used shape to be assigned to this frame, expressed in frame coordinates, or kODNULL if the used shape is the same as the frame shape.

---

## ChangeUsedShape Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in platform-normal coordinates.

---

## ChangeUsedShape - Return Value

None.

---

## ChangeUsedShape - Parameters

**shape** (ODShape \*) - input

A reference to a used shape to be assigned to this frame, expressed in frame coordinates, or kODNULL if the used shape is the same as the frame shape.

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in platform-normal coordinates.

None.

---

## ChangeUsedShape - Remarks

Your part assigns a used shape to its display frame by calling the frame's ChangeUsedShape method. This method in turn calls the [UsedShapeChanged](#) method of the display frame's containing part to notify the containing part that its used shape has changed.

---

## ChangeUsedShape - Related Methods

### Related Methods

- [ODFrame::AcquireUsedShape](#)
- [ODPart::UsedShapeChanged](#)



---

## ChangeUsedShape - Topics

### Class:

ODFrame

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

## ChangeViewType

---

## ChangeViewType - Syntax

This method changes the specified view type of this frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>
```

```
ODTypeToken    viewType;
```

```
ChangeViewType(viewType);
```

---

## ChangeViewType Parameter - viewType

**viewType** ([ODTypeToken](#)) - input

A tokenized string representing the view type to be assigned to this frame.

This parameter must be the tokenized form of one of the following value-type constants. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODViewAsFrame	Frame view type.
kODViewAsLargeIcon	Large-icon (standard) view type.
kODViewAsSmallIcon	Small-icon view type.
kODViewAsThumbnail	Thumbnail view type.

---

## ChangeViewType - Return Value

None.

---

## ChangeViewType - Parameters

**viewType** ([ODTypeToken](#)) - input

A tokenized string representing the view type to be assigned to this frame.

This parameter must be the tokenized form of one of the following value-type constants. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODViewAsFrame

Frame view type.

kODViewAsLargeIcon

Large-icon (standard) view type.

kODViewAsSmallIcon

Small-icon view type.

kODViewAsThumbnail

Thumbnail view type.

None.

---

## ChangeViewType - Remarks

Your part calls its embedded frame's `ChangeViewType` method to request that it use a new view type. This method in turn calls the [ViewTypeChanged](#) method of this frame's part to notify the part that its view type has changed. If the embedded part does not support the requested view type, it can pick a view type that it can support and then call its frame's [SetViewType](#) method to update the view type in the frame.

---

## ChangeViewType - Exception Handling

kODErrIllegalNullTokenInput

The *viewType* parameter is null.

---

## ChangeViewType - Related Methods

### Related Methods

- [ODFrame::GetViewType](#)
  - [ODFrame::SetViewType](#)
  - [ODPart::ViewTypeChanged](#)
  - [ODSession::Tokenize](#)
-

# ChangeViewType - Topics

## Class:

ODFrame

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

## Close

---

## Close - Syntax

This method prepares this frame to be removed from memory but does not affect persistent storage.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>
```

```
Close();
```

---

## Close - Return Value

None.

---

## Close - Parameters

None.

---

## Close - Remarks

Your part calls its embedded frame's Close method when a document is being closed after its final save. This method in turn calls the [DisplayFrameClosed](#) method associated with this frame's part to notify the part that it is being closed.

During execution of this method, this frame releases its references to the associated part and its containing frame, and this frame should not be used again.

Depending on whether you want to affect persistent storage, your part calls either this method or its frame's [Remove](#) method.

---

## Close - Related Methods

### Related Methods

- [ODFrame::Remove](#)
- [ODPart::DisplayFrameClosed](#)

---

## Close - Topics

### Class:

[ODFrame](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

## ContentUpdated

---

## ContentUpdated - Syntax

This method notifies this frame's containing part that this frame's part has updated its content, so the containing part can update any link sources that it maintains.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>
```

```
ODUpdateID    change;
```

```
ContentUpdated(change);
```

---

## ContentUpdated Parameter - change

**change** ([ODUpdateID](#)) - input  
The update ID associated with this frame.

---

## ContentUpdated - Return Value

None.

---

## ContentUpdated - Parameters

**change** ([ODUpdateID](#)) - input  
The update ID associated with this frame.

None.

---

## ContentUpdated - Remarks

Your part calls its display frame's ContentUpdated method when your part's content changes. Your part should avoid calling this method at every content change; it generally calls this method after a reasonable pause and when the part loses the selection focus. This method may be called multiple times. This method in turn calls the [EmbeddedFrameUpdated](#) method for all containing parts in the frame hierarchy through the root part of the window, so that any affected link source can be updated.

---

## ContentUpdated - Related Methods

### Related Methods

- [ODPart::EmbeddedFrameUpdated](#)
- 

## ContentUpdated - Topics

**Class:**  
ODFrame

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

# CreateFacetIterator

---

## CreateFacetIterator - Syntax

This method creates a frame-facet iterator object for the facets of this frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODFrameFacetIterator    *rv;

rv = CreateFacetIterator();
```

---

## CreateFacetIterator Return Value - rv

**rv** (ODFrameFacetIterator \*) - returns  
A reference to a new frame-facet iterator object.

---

## CreateFacetIterator - Parameters

**rv** (ODFrameFacetIterator \*) - returns  
A reference to a new frame-facet iterator object.

---

## CreateFacetIterator - Remarks

Your part calls this method if it needs to apply an operation to all facets of one of your display frames, such as drawing and changing their shapes. It is your responsibility to delete the iterator when it is no longer needed.

While you are using a frame-facet iterator, you should not modify the list of facets for the frame. You must postpone adding items to or removing items from the list of facets for the frame until after you have deleted the iterator.

---

## CreateFacetIterator - Related Methods

**Related Methods**

- [ODFacet::CreateFacetIterator](#)

---

## CreateFacetIterator - Topics

### Class:

ODFrame

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## CreateShape

---

## CreateShape - Syntax

This method creates a shape object.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODShape      *rv;

rv = CreateShape();
```

---

## CreateShape Return Value - rv

**rv** (ODShape \*) - returns  
A reference to a new shape object.

---

## CreateShape - Parameters

**rv** (ODShape \*) - returns  
A reference to a new shape object.

---

## CreateShape - Remarks

Your part calls this method to create a shape object for any purpose.

This method initializes the reference count of the returned shape object. When you have finished using that shape object, you should call its [Release](#) method.

---

## CreateShape - Related Methods

### Related Methods

- [ODFacet::CreateShape](#)
- 

## CreateShape - Topics

### Class:

ODFrame

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## CreateTransform

---

## CreateTransform - Syntax

This method creates a transform object.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODTransform      *rv;

rv = CreateTransform();
```

---

## CreateTransform Return Value - rv



**rv** (ODTransform \*) - returns  
A reference to a new transform object.

---

## CreateTransform - Parameters

**rv** (ODTransform \*) - returns  
A reference to a new transform object.

---

## CreateTransform - Remarks

Your part calls this method to create a transform object for any purpose.

This method initializes the reference count of the returned transform object. When you have finished using that transform object, you should call its [Release](#) method.

---

## CreateTransform - Related Methods

### Related Methods

- [ODFacet::CreateTransform](#)
- 

## CreateTransform - Topics

**Class:**  
ODFrame

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## DoesPropagateEvents

---

## DoesPropagateEvents - Syntax

This method indicates whether this frame propagates unhandled events to its containing part.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODBoolean    rv;

rv = DoesPropagateEvents();
```

---

## DoesPropagateEvents Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating whether this frame propagates unhandled events to its containing part.

kODTrue

This frame propagates unhandled events to its containing part.

kODFalse

This frame does not propagate unhandled events to its containing part.

---

## DoesPropagateEvents - Parameters

**rv** ([ODBoolean](#)) - returns

A flag indicating whether this frame propagates unhandled events to its containing part.

kODTrue

This frame propagates unhandled events to its containing part.

kODFalse

This frame does not propagate unhandled events to its containing part.

---

## DoesPropagateEvents - Remarks

If one of your part's embedded frames propagates unhandled events to your part, your part has the opportunity to handle those events. In that case, your part's event handler needs to determine if a particular frame is one of its display frames or an embedded frame.

---

## DoesPropagateEvents - Related Methods

### Related Methods

- [ODFrame::SetPropagateEvents](#)
- 

## DoesPropagateEvents - Topics

**Class:**  
ODFrame

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## DrawActiveBorder

---

### DrawActiveBorder - Syntax

This method updates the active frame border of this frame.

```
#define INCL_ODFRAME  
#define INCL_ODAPI  
#include <os2.h>
```

```
DrawActiveBorder();
```

---

### DrawActiveBorder - Return Value

None.

---

### DrawActiveBorder - Parameters

None.

---

### DrawActiveBorder - Remarks

OpenDoc calls this method. This method in turn calls the [DrawActiveBorder](#) method associated with each facet of this frame.

---

# DrawActiveBorder - Related Methods

## Related Methods

- [ODFacet::DrawActiveBorder](#)
- [ODFrame::InvalidateActiveBorder](#)

---

# DrawActiveBorder - Topics

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

# EditInLink

---

## EditInLink - Syntax

This method indicates whether the part maintaining a link destination that includes this frame can be found.

```
#define INCL_ODFRAME  
#define INCL_ODAPI  
#include <os2.h>
```

```
ODBoolean    rv;  
  
rv = EditInLink();
```

---

## EditInLink Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the part maintaining the link destination that includes this frame can be found.

kODTrue	The part maintaining the link destination can be found.
kODFalse	The part maintaining the link destination cannot be found.

---

## EditInLink - Parameters

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the part maintaining the link destination that includes this frame can be found.

kODTrue

The part maintaining the link destination can be found.

kODFalse

The part maintaining the link destination cannot be found.

-----

## EditInLink - Remarks

Your part calls this method when the user attempts to edit in a display frame which has link status of kODInLinkDestination. This method in turn calls the [EditInLinkAttempted](#) method of the part that maintains that link destination. The part then displays an alert allowing the user to find the source of the link or to break the link. If the user chooses to break the link, that part changes the link status of the embedded frame accordingly.

If the part maintaining the link destination was found, this method returns kODTrue, and if the link status of this frame was changed, you can then allow editing. In the unlikely event that the part maintaining the link destination cannot be found, this method returns kODFalse, and your part should display an alert informing the user that the destination of the link cannot be edited.

-----

## EditInLink - Related Methods

### Related Methods

- [ODPart::EditInLinkAttempted](#)

-----

## EditInLink - Topics

### Class:

ODFrame

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

-----

## FacetAdded

-----

## FacetAdded - Syntax

This method adds the facet to this frames's list of facets and notifies its part of the new facet.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODFacet      *facet;

FacetAdded(facet);
```

---

## FacetAdded Parameter - facet

**facet** (ODFacet \*) - input  
A reference to a facet to be added.

---

## FacetAdded - Return Value

None.

---

## FacetAdded - Parameters

**facet** (ODFacet \*) - input  
A reference to a facet to be added.

None.

---

## FacetAdded - Remarks

OpenDoc calls this method when a facet is added to this frame. This method in turn calls the [FacetAdded](#) method associated with this frame's part to notify the part that a facet has been added to one of its display frames.

---

## FacetAdded - Related Methods

### Related Methods

- [ODFrame::FacetRemoved](#)
  - [ODPart::FacetAdded](#)
-

# FacetAdded - Topics

## Class:

ODFrame

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

## FacetRemoved

---

## FacetRemoved - Syntax

This method removes the facet from this frame's list of facets and notifies its part of the deletion.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODFacet      *facet;

FacetRemoved(facet);
```

---

## FacetRemoved Parameter - facet

**facet** (ODFacet \*) - input  
A reference to a facet to be removed.

---

## FacetRemoved - Return Value

None.

---

## FacetRemoved - Parameters

**facet** (ODFacet \*) - input  
A reference to a facet to be removed.

None.

---

## FacetRemoved - Remarks

OpenDoc calls this method when a facet is removed. This method in turn calls the [FacetRemoved](#) method associated with this frame's part to notify the part that a facet has been removed from one of its display frames.

---

## FacetRemoved - Related Methods

### Related Methods

- [ODFacet::RemoveFacet](#)
  - [ODFrame::FacetAdded](#)
  - [ODPart::FacetRemoved](#)
- 

## FacetRemoved - Topics

### Class:

ODFrame

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## GetContentExtent

---

## GetContentExtent - Syntax

This method retrieves the content extent of this frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>
```



```
ODPoint *contentExtent;  
GetContentExtent (contentExtent);
```

---

## GetContentExtent Parameter - contentExtent

**contentExtent** ([ODPoint \\*](#)) - output  
The content extent of this frame.

---

## GetContentExtent - Return Value

None.

---

## GetContentExtent - Parameters

**contentExtent** ([ODPoint \\*](#)) - output  
The content extent of this frame.

None.

---

## GetContentExtent - Remarks

The content extent is, in essence, the page height of the part displayed in this frame.

---

## GetContentExtent - Related Methods

### Related Methods

- [ODFrame::ChangeContentExtent](#)
- 

## GetContentExtent - Topics

Class:

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

## GetFrameGroup

---

### GetFrameGroup - Syntax

This method returns the group ID of this frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>
```

```
ODULong    rv;
```

```
rv = GetFrameGroup();
```

---

### GetFrameGroup Return Value - rv

**rv** ([ODULong](#)) - returns  
The group ID of this frame.

---

### GetFrameGroup - Parameters

**rv** ([ODULong](#)) - returns  
The group ID of this frame.

---

### GetFrameGroup - Related Methods

#### Related Methods

- [ODFrame::SetFrameGroup](#)

---

# GetFrameGroup - Topics

## Class:

ODFrame

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Related Methods](#)

---

## GetLinkStatus

---

## GetLinkStatus - Syntax

This method returns the link status of this frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>
```

```
ODLinkStatus    rv;
```

```
rv = GetLinkStatus();
```

---

## GetLinkStatus Return Value - rv

**rv** ([ODLinkStatus](#)) - returns

The link status of this frame. This parameter returns one of the following values:

kODInLinkDestination

The frame is embedded in the destination of a link; the content of this frame is thus supplied by a link.

kODInLinkSource

The frame is embedded in the content that is the source of one or more links, but not in content that is the destination of a link.

kODNotInLink

The frame is not embedded in any linked content, source or destination.

---

## GetLinkStatus - Parameters

**rv** ([ODLinkStatus](#)) - returns

The link status of this frame. This parameter returns one of the following values:

**kODInLinkDestination**

The frame is embedded in the destination of a link; the content of this frame is thus supplied by a link.

**kODInLinkSource**

The frame is embedded in the content that is the source of one or more links, but not in content that is the destination of a link.

**kODNotInLink**

The frame is not embedded in any linked content, source or destination.

-----

## GetLinkStatus - Remarks

Your part calls its display frame's `GetLinkStatus` method to determine if it should allow links to be created to or from the content displayed by this frame. For example, if the link status is `kODInLinkDestination`, a link should not be created within this frame.

-----

## GetLinkStatus - Related Methods

### Related Methods

- [ODFrame::ChangeLinkStatus](#)

-----

## GetLinkStatus - Topics

### Class:

`ODFrame`

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

-----

## GetPartInfo

-----

## GetPartInfo - Syntax

This method returns the part-information data for this frame.

```
#define INCL_ODFRAME
```

```
#define INCL_ODAPI
#include <os2.h>

ODInfoType    rv;

rv = GetPartInfo();
```

-----

## GetPartInfo Return Value - rv

**rv** ([ODInfoType](#)) - returns  
The part-information data for this frame.

-----

## GetPartInfo - Parameters

**rv** ([ODInfoType](#)) - returns  
The part-information data for this frame.

-----

## GetPartInfo - Remarks

Your part calls its display frame's GetPartInfo method to retrieve any part-specific information it has stored there.

You should cast the return value to a pointer to your part's own representation of the data.

-----

## GetPartInfo - Related Methods

### Related Methods

- [ODFacet::GetPartInfo](#)
- [ODFrame::SetPartInfo](#)

-----

## GetPartInfo - Topics

**Class:**  
ODFrame

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

# GetPresentation

---

## GetPresentation - Syntax

This method returns the presentation of this frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODTypeToken    rv;

rv = GetPresentation();
```

---

## GetPresentation Return Value - rv

**rv** ([ODTypeToken](#)) - returns  
A tokenized string representing this frame's presentation.

---

## GetPresentation - Parameters

**rv** ([ODTypeToken](#)) - returns  
A tokenized string representing this frame's presentation.

---

## GetPresentation - Related Methods

### Related Methods

- [ODFrame::ChangePresentation](#)
  - [ODFrame::SetPresentation](#)
- 

## GetPresentation - Topics

**Class:**

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Related Methods](#)

---

# GetSequenceNumber

---

## GetSequenceNumber - Syntax

This method returns the sequence number of this frame in its frame group.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODULong    rv;

rv = GetSequenceNumber();
```

---

## GetSequenceNumber Return Value - rv

**rv** ([ODULong](#)) - returns  
The sequence number of this frame.

---

## GetSequenceNumber - Parameters

**rv** ([ODULong](#)) - returns  
The sequence number of this frame.

---

## GetSequenceNumber - Related Methods

### Related Methods

- [ODFrame::ChangeSequenceNumber](#)
-

# GetSequenceNumber - Topics

## Class:

ODFrame

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Related Methods](#)

---

## GetViewType

---

## GetViewType - Syntax

This method returns the view type of this frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>
```

```
ODTypeToken    rv;
```

```
rv = GetViewType();
```

---

## GetViewType Return Value - rv

**rv** ([ODTypeToken](#)) - returns

A tokenized string representing the view type of this frame.

This parameter must be the tokenized form off one of the following view-type constants. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODViewAsFrame	Frame view type.
kODViewAsLargeIcon	Large-icon (standard) view type.
kODViewAsSmallIcon	Small-icon view type.
kODViewAsThumbnail	Thumbnail view type.

---

## GetViewType - Parameters



**rv** ([ODTypeToken](#)) - returns

A tokenized string representing the view type of this frame.

This parameter must be the tokenized form off one of the following view-type constants. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

**kODViewAsFrame**  
Frame view type.  
**kODViewAsLargelcon**  
Large-icon (standard) view type.  
**kODViewAsSmalllcon**  
Small-icon view type.  
**kODViewAsThumbnail**  
Thumbnail view type.

---

## GetViewType - Related Methods

### Related Methods

- [ODFrame::ChangeViewType](#)
  - [ODFrame::SetViewType](#)
  - [ODSession::Tokenize](#)
- 

## GetViewType - Topics

### Class:

[ODFrame](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Related Methods](#)

---

## Invalidate

---

## Invalidate - Syntax

This method marks the specified area in this frame as in need of updating.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODShape      *invalidShape;
ODCanvas     *biasCanvas;

Invalidate(invalidShape, biasCanvas);
```

---

## Invalidate Parameter - invalidShape

**invalidShape** (ODShape \*) - input

A reference to the shape object defining the area in this frame that needs updating, expressed in frame coordinates.

---

## Invalidate Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in platform-normal coordinates.

---

## Invalidate - Return Value

None.

---

## Invalidate - Parameters

**invalidShape** (ODShape \*) - input

A reference to the shape object defining the area in this frame that needs updating, expressed in frame coordinates.

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in platform-normal coordinates.

None.

---

## Invalidate - Remarks

Your part calls this method to explicitly invalidate a portion of its display frame. This method in turn calls the [Invalidate](#) method associated with each of this frame's facets. The resulting invalid shape is transformed and clipped to the coordinate space of each facet's canvas.

---

## Invalidate - Related Methods

**Related Methods**

- [ODCanvas::Invalidate](#)
- [ODFacet::Invalidate](#)
- [ODFrame::Validate](#)

---

## Invalidate - Topics

### Class:

ODFrame

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

## InvalidateActiveBorder

---

## InvalidateActiveBorder - Syntax

This method marks the active frame border of this frame as in need of updating.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>
```

```
InvalidateActiveBorder();
```

---

## InvalidateActiveBorder - Return Value

None.

---

## InvalidateActiveBorder - Parameters

None.

---

## InvalidateActiveBorder - Remarks

OpenDoc calls this method. This method in turn calls the [InvalidateActiveBorder](#) method of each facet of this frame.

---

## InvalidateActiveBorder - Related Methods

### Related Methods

- [ODFacet::InvalidateActiveBorder](#)
  - [ODFrame::DrawActiveBorder](#)
- 

## InvalidateActiveBorder - Topics

### Class:

ODFrame

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## IsDragging

---

## IsDragging - Syntax

This method indicates whether this frame is currently being dragged.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    rv;
```

```
rv = IsDragging();
```

---

## IsDragging Return Value - rv

**rv** ([ODBoolean](#)) - returns  
A flag indicting whether this frame is currently being dragged.

kODTrue	This frame is currently being dragged.
kODFalse	This frame is not currently being dragged.

---

## IsDragging - Parameters

**rv** ([ODBoolean](#)) - returns  
A flag indicting whether this frame is currently being dragged.

kODTrue	This frame is currently being dragged.
kODFalse	This frame is not currently being dragged.

---

## IsDragging - Remarks

OpenDoc calls this method to determine whether parts can be dropped onto this frame.

---

## IsDragging - Related Methods

### Related Methods

- [ODFrame::IsDroppable](#)
  - [ODFrame::SetDragging](#)
  - [ODFrame::SetDroppable](#)
- 

## IsDragging - Topics

### Class:

ODFrame

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## IsDroppable

---

## IsDroppable - Syntax

This method indicates whether this frame's part accepts drag-and-drop events in this frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODBoolean    rv;

rv = IsDroppable();
```

---

## IsDroppable Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating whether this frame's part accepts drag-and-drop events in this frame.

kODTrue

This frame's part accepts drag-and-drop events in this frame.

kODFalse

This frame's part does not accept drag-and-drop events in this frame.

---

## IsDroppable - Parameters

**rv** ([ODBoolean](#)) - returns

A flag indicating whether this frame's part accepts drag-and-drop events in this frame.

kODTrue

This frame's part accepts drag-and-drop events in this frame.

kODFalse

This frame's part does not accept drag-and-drop events in this frame.

---

## IsDroppable - Remarks

OpenDoc calls this method before calling your part's [DragEnter](#) method to ensure that your part's display frame is able to receive a drop.

---

## IsDroppable - Related Methods

**Related Methods**

- [ODFrame::IsDragging](#)
- [ODFrame::SetDragging](#)
- [ODFrame::SetDroppable](#)

---

## IsDroppable - Topics

### Class:

[ODFrame](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

## IsFrozen

---

## IsFrozen - Syntax

This method indicates whether this frame is bundled.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    rv;
```

```
rv = IsFrozen();
```

---

## IsFrozen Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating whether this frame is bundled.

kODTrue

This frame is bundled.

kODFalse

This frame is not bundled.

---

## IsFrozen - Parameters

**rv** ([ODBoolean](#)) - returns  
A flag indicating whether this frame is bundled.

kODTrue	This frame is bundled.
kODFalse	This frame is not bundled.

---

## IsFrozen - Related Methods

### Related Methods

- [ODFrame::SetFrozen](#)

---

## IsFrozen - Topics

### Class:

ODFrame

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Related Methods](#)

---

## IsInLimbo

---

## IsInLimbo - Syntax

This method indicates whether this frame was removed from a part's content model.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    rv;
```

```
rv = IsInLimbo();
```

---

## IsInLimbo Return Value - rv



**rv** ([ODBoolean](#)) - returns

A flag indicating whether this frame was removed from a part's content model.

kODTrue

This frame was removed from a part's content model.

kODFalse

This frame was in a part's content model (such as an embedded frame).

---

## IsInLimbo - Parameters

**rv** ([ODBoolean](#)) - returns

A flag indicating whether this frame was removed from a part's content model.

kODTrue

This frame was removed from a part's content model.

kODFalse

This frame was in a part's content model (such as an embedded frame).

---

## IsInLimbo - Remarks

Your part calls this method to check whether the frame is in limbo.

The in-limbo flag is not a persistent property of frames; frames that are in limbo are eventually removed. When a frame object is created by a draft, the frame is not in limbo. If this frame was removed from a part's content model, such as by the cut operation, this frame is in limbo, and eventually this frame is removed from the draft by the part that last contained the frame in its content model. If this frame is inserted into a part's content model, such as by a paste or undo of the cut operation, this frame is again not in limbo.

---

## IsInLimbo - Related Methods

### Related Methods

- [ODFrame::SetInLimbo](#)
- 

## IsInLimbo - Topics

### Class:

[ODFrame](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

# IsOverlaid

---

## IsOverlaid - Syntax

This method indicates whether this frame is an overlaid frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    rv;

rv = IsOverlaid();
```

---

## IsOverlaid Return Value - rv

**rv** (ODBoolean) - returns

A flag indicating whether this frame is an overlaid frame.

kODTrue	This frame is an overlaid frame.
kODFalse	This frame is not an overlaid frame.

---

## IsOverlaid - Parameters

**rv** (ODBoolean) - returns

A flag indicating whether this frame is an overlaid frame.

kODTrue	This frame is an overlaid frame.
kODFalse	This frame is not an overlaid frame.

---

## IsOverlaid - Remarks

This method's return value is defined for the lifetime of this frame; once set, it cannot be changed.

---

## IsOverlaid - Related Methods

## Related Methods

- [ODFrame::IsRoot](#)
- [ODFrame::IsSubframe](#)

---

# IsOverlaid - Topics

## Class:

ODFrame

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

# IsRoot

---

## IsRoot - Syntax

This method indicates whether this frame is the root frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    rv;
```

```
rv = IsRoot();
```

---

## IsRoot Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating whether this frame is the root frame.

kODTrue

This frame is the root frame.

kODFalse

This frame is not the root frame.

---

## IsRoot - Parameters

**rv** ([ODBoolean](#)) - returns  
A flag indicating whether this frame is the root frame.

kODTrue	This frame is the root frame.
kODFalse	This frame is not the root frame.

---

## IsRoot - Remarks

This method's return value is defined for the lifetime of this frame; once set, it cannot be changed.

---

## IsRoot - Related Methods

### Related Methods

- [ODFrame::IsSubframe](#)
  - [ODFrame::IsOverlaid](#)
- 

## IsRoot - Topics

**Class:**  
[ODFrame](#)

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## IsSubframe

---

## IsSubframe - Syntax

This method indicates whether this frame is a subframe of its containing frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean rv;  
  
rv = IsSubframe();
```

---

## IsSubframe Return Value - rv

**rv** ([ODBoolean](#)) - returns  
A flag indicating whether this frame is a subframe of its containing frame.

kODTrue	This frame is a subframe of its containing frame.
kODFalse	This frame is not a subframe of its containing frame.

---

## IsSubframe - Parameters

**rv** ([ODBoolean](#)) - returns  
A flag indicating whether this frame is a subframe of its containing frame.

kODTrue	This frame is a subframe of its containing frame.
kODFalse	This frame is not a subframe of its containing frame.

---

## IsSubframe - Related Methods

### Related Methods

- [ODFrame::IsRoot](#)
- [ODFrame::SetSubframe](#)

---

## IsSubframe - Topics

**Class:**  
[ODFrame](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Related Methods](#)

---

Remove

---

## Remove - Syntax

This method prepares this frame to be removed from both memory and persistent storage.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>
```

```
Remove ();
```

---

## Remove - Return Value

None.

---

## Remove - Parameters

None.

---

## Remove - Remarks

Your part calls its embedded frames's `Remove` method when it permanently removes the frame from its content. This method in turn calls the [DisplayFrameRemoved](#) method of this frame's part to notify the part that this frame is being removed. During the execution of the [DisplayFrameRemoved](#) method, this frame's part calls its embedded frames' `Remove` method recursively.

During execution of this method, this frame releases its references to the associated part and its containing frame, and this frame should not be used again. After this method executes successfully, you do not need to call your part's [Release](#) method.

Depending on whether you want to affect persistent storage, your part calls either this method or its frame's [Close](#) method.

---

## Remove - Related Methods

### Related Methods

- [ODFrame::Close](#)
- [ODPart::DisplayFrameRemoved](#)

---

## Remove - Topics

### Class:

ODFrame

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

## RequestFrameShape

---

## RequestFrameShape - Syntax

This method requests a new frame shape for this frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODShape      *shape;
ODCanvas     *biasCanvas;
ODShape      *rv;

rv = RequestFrameShape(shape, biasCanvas);
```

---

## RequestFrameShape Parameter - shape

**shape** (ODShape \*) - input

A reference to the requested shape, expressed in frame coordinates.

---

## RequestFrameShape Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or KODNULL if the geometry is in platform-normal coordinates.

---

# RequestFrameShape Return Value - rv

**rv** (ODShape \*) - returns  
A reference to the new frame shape, expressed in frame coordinates.

---

## RequestFrameShape - Parameters

**shape** (ODShape \*) - input  
A reference to the requested shape, expressed in frame coordinates.

**biasCanvas** (ODCanvas \*) - input  
A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in platform-normal coordinates.

**rv** (ODShape \*) - returns  
A reference to the new frame shape, expressed in frame coordinates.

---

## RequestFrameShape - Remarks

Your part calls its display frame's RequestFrameShape method when it wants to resize the display frame. This method in turn calls the [RequestFrameShape](#) method of this frame's containing part. The containing part returns a reference to the shape object it allows the display frame to have. This frame stores the shape as its new frame shape and returns the shape to your part so that your part knows what the new shape is.

Your part must not modify the returned shape. Only this frame's containing part can modify the frame shape.

This method increments the reference count of the returned shape object. When you have finished using that shape object, you should call its [Release](#) method.

---

## RequestFrameShape - Exception Handling

kODErrIllegalNullShapeInput	The <i>shape</i> parameter is null.
-----------------------------	-------------------------------------

---

## RequestFrameShape - Related Methods

### Related Methods

- [ODFrame::ChangeFrameShape](#)
  - [ODFrame::AcquireFrameShape](#)
  - [ODPart::RequestFrameShape](#)
- 

## RequestFrameShape - Topics



**Class:**

ODFrame

Select an item:

[Syntax](#)[Parameters](#)[Returns](#)[Remarks](#)[Exception Handling](#)[Related Methods](#)

---

## SetContainingFrame

---

### SetContainingFrame - Syntax

This method assigns the specified frame as the containing frame of this frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODFrame      *frame;

SetContainingFrame(frame);
```

---

### SetContainingFrame Parameter - frame

**frame** (ODFrame \*) - input

A reference to the frame to be assigned as this frame's containing frame.

---

### SetContainingFrame - Return Value

None.

---

### SetContainingFrame - Parameters

**frame** (ODFrame \*) - input

A reference to the frame to be assigned as this frame's containing frame.

None.

-----

## SetContainingFrame - Remarks

You should remove any existing facets of this frame before calling this method. It is necessary to call this method only when a frame is moved because the frame is always created with the correct containing frame.

-----

## SetContainingFrame - Related Methods

### Related Methods

- [ODFrame::AcquireContainingFrame](#)

-----

## SetContainingFrame - Topics

### Class:

ODFrame

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

-----

## SetDragging

-----

## SetDragging - Syntax

This method specifies whether this frame is currently being dragged.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODBoolean    isDragging;

SetDragging(isDragging);
```

---

## SetDragging Parameter - isDragging

**isDragging** ([ODBoolean](#)) - input

A flag indicating whether this frame is currently being dragged.

kODTrue

This frame is currently being dragged.

kODFalse

This frame is not currently being dragged.

---

## SetDragging - Return Value

None.

---

## SetDragging - Parameters

**isDragging** ([ODBoolean](#)) - input

A flag indicating whether this frame is currently being dragged.

kODTrue

This frame is currently being dragged.

kODFalse

This frame is not currently being dragged.

None.

---

## SetDragging - Remarks

Your part calls this method to indicate whether parts can be dropped on to this frame.

---

## SetDragging - Related Methods

### Related Methods

- [ODFrame::IsDragging](#)
  - [ODFrame::IsDroppable](#)
  - [ODFrame::SetDroppable](#)
-

# SetDragging - Topics

## Class:

ODFrame

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

## SetDroppable

---

## SetDroppable - Syntax

This method specifies whether this frame's part accepts drag-and-drop events in this frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    isDroppable;
```

```
SetDroppable(isDroppable);
```

---

## SetDroppable Parameter - isDroppable

**isDroppable** ([ODBoolean](#)) - input

A flag indicating whether this frame's part accepts drag-and-drop events in this frame.

kODTrue

This frame's part accepts drag-and-drop events in this frame.

kODFalse

This frame's part does not accept drag-and-drop events in this frame.

---

## SetDroppable - Return Value

None.

# SetDroppable - Parameters

**isDroppable** ([ODBoolean](#)) - input

A flag indicating whether this frame's part accepts drag-and-drop events in this frame.

kODTrue

This frame's part accepts drag-and-drop events in this frame.

kODFalse

This frame's part does not accept drag-and-drop events in this frame.

None.

---

## SetDroppable - Remarks

Your part calls this method to define its display frame as either capable or incapable of accepting a drop.

---

## SetDroppable - Related Methods

### Related Methods

- [ODFrame::IsDragging](#)
  - [ODFrame::IsDroppable](#)
  - [ODFrame::SetDragging](#)
- 

## SetDroppable - Topics

### Class:

[ODFrame](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## SetFrameGroup

---

## SetFrameGroup - Syntax

This method assigns a group ID to this frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODULong    groupID;

SetFrameGroup(groupID);
```

---

## SetFrameGroup Parameter - groupID

**groupID** ([ODULong](#)) - input  
The group ID to be assigned to this frame.

---

## SetFrameGroup - Return Value

None.

---

## SetFrameGroup - Parameters

**groupID** ([ODULong](#)) - input  
The group ID to be assigned to this frame.

None.

---

## SetFrameGroup - Remarks

Your part calls its embedded frame's SetFrameGroup method to change its group ID.

---

## SetFrameGroup - Related Methods

### Related Methods

- [ODFrame::GetFrameGroup](#)

# SetFrameGroup - Topics

## Class:

ODFrame

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

## SetFrozen

---

## SetFrozen - Syntax

This method specifies whether this frame is currently bundled.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    isFrozen;
```

```
SetFrozen(isFrozen);
```

---

## SetFrozen Parameter - isFrozen

**isFrozen** ([ODBoolean](#)) - input

A flag indicating whether this frame is currently bundled.

kODTrue

This frame is currently bundled.

kODFalse

This frame is not currently bundled.

---

## SetFrozen - Return Value

None.

# SetFrozen - Parameters

**isFrozen** ([ODBoolean](#)) - input

A flag indicating whether this frame is currently bundled.

kODTrue

This frame is currently bundled.

kODFalse

This frame is not currently bundled.

None.

---

## SetFrozen - Remarks

Your part calls this method for any embedded frame or display frame.

---

## SetFrozen - Related Methods

### Related Methods

- [ODFrame::IsFrozen](#)
- 

## SetFrozen - Topics

### Class:

ODFrame

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

## SetInLimbo

---

## SetInLimbo - Syntax

This method specifies whether this frame is to be removed from a part's content model.



```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODBoolean    isInLimbo;

SetInLimbo(isInLimbo);
```

-----

## SetInLimbo Parameter - isInLimbo

**isInLimbo** (ODBoolean) - input

A flag indicating whether this frame is to be removed from a part's content model.

kODTrue

This frame is to be removed from a part's content model.

kODFalse

This frame is currently in a part's content model (for example, an embedded frame).

-----

## SetInLimbo - Return Value

None.

-----

## SetInLimbo - Parameters

**isInLimbo** (ODBoolean) - input

A flag indicating whether this frame is to be removed from a part's content model.

kODTrue

This frame is to be removed from a part's content model.

kODFalse

This frame is currently in a part's content model (for example, an embedded frame).

None.

-----

## SetInLimbo - Remarks

Your part calls this method to specify whether this frame is in limbo.

The in-limbo flag is not a persistent property of frames; frames that are in limbo are eventually removed. If the frame is in limbo, this frame is removed from the draft by the part that last contained the frame in its content model. This method does not need to recursively set the flag of embedded frames.

-----

# SetInLimbo - Related Methods

## Related Methods

- [ODFrame::IsInLimbo](#)

---

## SetInLimbo - Topics

### Class:

[ODFrame](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

## SetPartInfo

---

## SetPartInfo - Syntax

This method assigns part-information data to this frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODInfoType    partInfo;

SetPartInfo(partInfo);
```

---

## SetPartInfo Parameter - partInfo

**partInfo** ([ODInfoType](#)) - input

The part-information data to be assigned to this frame.

---

## SetPartInfo - Return Value

None.

---

## SetPartInfo - Parameters

**partInfo** ([ODInfoType](#)) - input

The part-information data to be assigned to this frame.

None.

---

## SetPartInfo - Remarks

Your part calls its display frame's SetPartInfo method.

---

## SetPartInfo - Related Methods

### Related Methods

- [ODFacet::SetPartInfo](#)
  - [ODFrame::GetPartInfo](#)
- 

## SetPartInfo - Topics

### Class:

[ODFrame](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## SetPresentation

---

## SetPresentation - Syntax

This method assigns the specified presentation to this frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODTypeToken    presentation;

SetPresentation(presentation);
```

---

## SetPresentation Parameter - presentation

**presentation** (ODTypeToken) - input

A tokenized string representing the presentation to be assigned to this frame. The presentation values are determined by the part developer. They must be tokenized to ensure they are unique. Non-unique or null values are invalid.

---

## SetPresentation - Return Value

None.

---

## SetPresentation - Parameters

**presentation** (ODTypeToken) - input

A tokenized string representing the presentation to be assigned to this frame. The presentation values are determined by the part developer. They must be tokenized to ensure they are unique. Non-unique or null values are invalid.

None.

---

## SetPresentation - Remarks

Your part calls its display frame's SetPresentation method.

---

## SetPresentation - Exception Handling

kODErrIllegalNullTokenInput

The *presentation* parameter is null.

---

## SetPresentation - Related Methods

### Related Methods

- [ODFrame::ChangePresentation](#)
  - [ODFrame::GetPresentation](#)
- 

## SetPresentation - Topics

### Class:

[ODFrame](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

## SetPropagateEvents

---

## SetPropagateEvents - Syntax

This method specifies whether this frame should propagate unhandled events to its containing frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODBoolean    doesPropagateEvents;

SetPropagateEvents(doesPropagateEvents);
```

---

## SetPropagateEvents Parameter - doesPropagateEvents

**doesPropagateEvents** ([ODBoolean](#)) - input

A flag indicating whether this frame should propagate unhandled events to its containing frame.

kODTrue

This frame should propagate unhandled events to its containing frame.

kODFalse

## SetPropagateEvents - Return Value

None.

---

## SetPropagateEvents - Parameters

**doesPropagateEvents** ([ODBoolean](#)) - input

A flag indicating whether this frame should propagate unhandled events to its containing frame.

kODTrue

This frame should propagate unhandled events to its containing frame.

kODFalse

This frame should not propagate unhandled events to its containing frame.

None.

---

## SetPropagateEvents - Remarks

Your part calls its embedded frame's SetPropagateEvents method to indicate whether it wants to receive events not handled by the embedded frame.

---

## SetPropagateEvents - Related Methods

### Related Methods

- [ODFrame::DoesPropagateEvents](#)
- 

## SetPropagateEvents - Topics

### Class:

ODFrame

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

# SetSubframe

---

## SetSubframe - Syntax

This method specifies whether this frame is currently a subframe of its containing frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODBoolean    isSubframe;

SetSubframe(isSubframe);
```

---

## SetSubframe Parameter - isSubframe

**isSubframe** (ODBoolean) - input

A flag indicating whether this frame is currently a subframe of a containing frame.

kODTrue

This frame is currently a subframe of a containing frame.

kODFalse

This frame is not currently a subframe of a containing frame.

---

## SetSubframe - Return Value

None.

---

## SetSubframe - Parameters

**isSubframe** (ODBoolean) - input

A flag indicating whether this frame is currently a subframe of a containing frame.

kODTrue

This frame is currently a subframe of a containing frame.

kODFalse

This frame is not currently a subframe of a containing frame.

None.

---

## SetSubframe - Related Methods

### Related Methods

- [ODFrame::IsSubframe](#)
- 

## SetSubframe - Topics

### Class:

ODFrame

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Related Methods](#)

---

## SetViewType

---

## SetViewType - Syntax

This method assigns the specified view type to this frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODTypeToken    viewType;

SetViewType(viewType);
```

---

## SetViewType Parameter - viewType

**viewType** ([ODTypeToken](#)) - input

A tokenized string representing the view type to be assigned to this frame.

This parameter must be the tokenized form of one of the following view-type constants. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODViewAsFrame



	Frame view type.
kODViewAsLargelcon	
	Large-icon (standard) view type.
kODViewAsSmallIcon	
	Small-icon view type.
kODViewAsThumbnail	
	Thumbnail view type.

-----

## SetViewType - Return Value

None.

-----

## SetViewType - Parameters

**viewType** ([ODTypeToken](#)) - input

A tokenized string representing the view type to be assigned to this frame.

This parameter must be the tokenized form of one of the following view-type constants. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODViewAsFrame	
	Frame view type.
kODViewAsLargelcon	
	Large-icon (standard) view type.
kODViewAsSmallIcon	
	Small-icon view type.
kODViewAsThumbnail	
	Thumbnail view type.

None.

-----

## SetViewType - Remarks

Your part calls its display frame's SetViewType method.

-----

## SetViewType - Exception Handling

kODErrIllegalNullTokenInput

The *viewType* parameter is null.

-----

## SetViewType - Related Methods

**Related Methods**

- [ODFrame::ChangeViewType](#)
  - [ODFrame::GetViewType](#)
  - [ODSession::Tokenize](#)
- 

## SetViewType - Topics

### Class:

ODFrame

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

## SetWindow

---

## SetWindow - Syntax

This method assigns a window to this frame.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>
```

```
ODWindow      *window;
```

```
SetWindow(window);
```

---

## SetWindow Parameter - window

**window** (ODWindow \*) - input

A reference to the window to be assigned to this frame.

---

## SetWindow - Return Value

None.

---

## SetWindow - Parameters

**window** (ODWindow \*) - input

A reference to the window to be assigned to this frame.

None.

---

## SetWindow - Remarks

OpenDoc calls this method when it creates this frame as its root frame. Only the root frame of a window has a reference to the window; all its embedded frames inherit this value.

---

## SetWindow - Exception Handling

kODErrNotRootFrame

The specified frame is not a root frame of this part.

---

## SetWindow - Related Methods

### Related Methods

- [ODFrame::AcquireWindow](#)
- 

## SetWindow - Topics

### Class:

ODFrame

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

## Validate

---

## Validate - Syntax

This method marks the specified area in this frame as no longer in need of updating.

```
#define INCL_ODFRAME
#define INCL_ODAPI
#include <os2.h>

ODShape      *validShape;
ODCanvas     *biasCanvas;

Validate(validShape, biasCanvas);
```

---

## Validate Parameter - validShape

**validShape** (ODShape \*) - input

A reference to the shape object defining the area in this frame that no longer needs updating, expressed in frame coordinates.

---

## Validate Parameter - biasCanvas

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in platform-normal coordinates.

---

## Validate - Return Value

None.

---

## Validate - Parameters

**validShape** (ODShape \*) - input

A reference to the shape object defining the area in this frame that no longer needs updating, expressed in frame coordinates.

**biasCanvas** (ODCanvas \*) - input

A reference to the canvas object to whose coordinate space the geometry is biased or kODNULL if the geometry is in platform-normal coordinates.

None.

---

## Validate - Remarks

Your part calls this method to explicitly mark portions of its display frame that do not need updating. This method in turn calls the [Validate](#) method associated with each of this frame's facets. The [Validate](#) method transforms and clips the shape from the coordinate space of the facet to the coordinate space of its canvas and subtracts the shape from any existing invalid area of the canvas.

---

## Validate - Related Methods

### Related Methods

- [ODCanvas::Validate](#)
  - [ODFacet::Validate](#)
  - [ODFrame::Invalidate](#)
- 

## Validate - Topics

### Class:

ODFrame

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## ODFrameFacetIterator

**Class Definition File:** FRFAITR.IDL

### Class Hierarchy

SOMObject  
ODObject  
    **ODFrameFacetIterator**

### Description

An object of the ODFrameFacetIterator class provides access to all facets of a frame.

A frame-facet iterator is used to apply an operation to all facets of one of your part's display frames, such as drawing and changing their shapes. For example, a part might use a frame-facet iterator to delete all facets of an embedded frame that it deletes or to update all views of one of its display frames if it were displaying asynchronously.

Your part creates a frame-facet iterator object by calling its frame's [CreateFacetIterator](#) method, which returns a reference to a frame-facet iterator object.

While you are using a frame-facet iterator, you should not modify the list of facets for the frame. You must postpone adding facets to or

removing facets from the list of facets for the frame until after you have deleted the iterator.

For more information related to facet objects, see the class description for [ODFacet](#). For more information on accessing objects through iterators, see the chapter on OpenDoc run-time features in the *OpenDoc Programming Guide* .

## Methods

The methods defined by the ODFacetIterator class include:

- [First](#)
- [IsNotComplete](#)
- [Next](#)

## Overridden Methods

There are currently no methods overridden by the ODFacetIterator class.

---

# First

---

## First - Syntax

This method begins the iteration and returns a reference to the first facet in the iteration sequence.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODFacet      *rv;

rv = First();
```

---

## First Return Value - rv

**rv** (ODFacet \*) - returns

A reference to the first facet in the iteration sequence or kODNULL if the frame has no facets.

---

## First - Parameters

**rv** (ODFacet \*) - returns

A reference to the first facet in the iteration sequence or kODNULL if the frame has no facets.

---

## First - Remarks

Your part must call this method before calling this frame-facet iterator's [IsNotComplete](#) method for the first time. This method can be called multiple times; each time, it resets the iteration.

---

## First - Exception Handling

kODErrIteratorOutOfSync

The list of facets for the frame was modified while the iteration was in progress.

---

## First - Topics

### Class:

ODFrameFacetIterator

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

---

## IsNotComplete

---

## IsNotComplete - Syntax

This method indicates whether the iteration is incomplete.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    rv;
```

```
rv = IsNotComplete();
```

---

## IsNotComplete Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the iteration is incomplete.

kODTrue

kODFalse	The iteration is incomplete.
	The iteration is complete.

---

## IsNotComplete - Parameters

**rv** ([ODBoolean](#)) - returns  
A flag indicating whether the iteration is incomplete.

kODTrue	The iteration is incomplete.
kODFalse	The iteration is complete.

---

## IsNotComplete - Remarks

Your part calls this method to test whether more facets remain in the iteration sequence. This method returns kODTrue if the preceding call to the [First](#) or [Next](#) method found a facet. This method returns kODFalse when you have examined all the facets (that is, when the previous call to [First](#) or [Next](#) returned kODNULL).

---

## IsNotComplete - Exception Handling

kODErrIteratorNotInitialized	This method was called before calling either the <a href="#">First</a> or <a href="#">Next</a> method to begin the iteration sequence.
kODErrIteratorOutOfSync	The list of facets for the frame was modified while the iteration was in progress.

---

## IsNotComplete - Topics

**Class:**  
[ODFrameFacetIterator](#)

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)

---

## Next



---

## Next - Syntax

This method returns a reference to the next facet in the iteration sequence.

```
#define INCL_ODFACET
#define INCL_ODAPI
#include <os2.h>

ODFacet      *rv;

rv = Next();
```

---

## Next Return Value - rv

**rv** (ODFacet \*) - returns

A reference to the next facet in the iteration sequence or KODNULL if you have reached the last facet.

---

## Next - Parameters

**rv** (ODFacet \*) - returns

A reference to the next facet in the iteration sequence or KODNULL if you have reached the last facet.

---

## Next - Remarks

If your part calls this method before calling this frame-facet iterator's [First](#) method to begin the iteration sequence, this method works the same as calling the [First](#) method.

---

## Next - Exception Handling

KODErrIterationOutOfSync

The list of facets for the frame was modified while the iteration was in progress.

---

## Next - Topics

**Class:**

ODFrameFacetIterator

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)

---

# ODHelp

**Class Definition File:** ODHELP.IDL

## Class Hierarchy

SOMObject  
  ODObject  
    **ODHelp**

## Description

Users can request help in a number of ways. They can select help from the **Help** menus, press the **F1** key while on a menu item, or press the **F1** key or select the **Help** button when they are in a control window of an active part. All parts should supply help for their applications both from the **Help** menu and from contextual help using the keyboard.

When you open a document, the document shell creates and initializes a single instance of ODHelp. All parts of that document share the help object; you can obtain a reference to it by calling the session object's [GetHelp](#) method.

OpenDoc forwards all help requests to the part for handling. If the part does not handle the request, OpenDoc tries to handle it. In the case of help on the document shell menu items, if the menu item is one of OpenDoc's predefined items, then a general help panel is shown for the action. If the menu item is defined by the part, but no help is provided by the part, OpenDoc displays a message that help is not available for that menu item. OpenDoc does not provide any default help for part-defined control windows or buttons. It is up to the part to display help panels for their part's contents.

The ODHelp class provides help developers an interface that allows parts to display help without creating and initializing the Help Manager help instance.

## Methods

The methods defined by the ODHelp class include:

- [DisplayHelp](#)
- [DisplayHelpIndex](#)
- [InitHelp](#)
- [TerminateHelp](#)

## Overridden Methods

There are currently no methods overridden by the ODHelp class.

---

# DisplayHelp (OS/2)

---

## DisplayHelp (OS/2) - Syntax

The method displays the help panel.

```
#define INCL_ODHELP
#define INCL_ODAPI
#include <os2.h>
```

```
string      sHelpFile;  
ODULong     ulPanelId;  
ODBoolean   rv;  
  
rv = DisplayHelp(sHelpFile, ulPanelId);
```

---

## DisplayHelp (OS/2) Parameter - sHelpFile

**sHelpFile** ([string](#)) - input  
The name of the IPF-compiled help file with an .HLP extension.

---

## DisplayHelp (OS/2) Parameter - ulPanelId

**ulPanelId** ([ODULong](#)) - input  
The ID of the help panel to be displayed.

---

## DisplayHelp (OS/2) Return Value - rv

**rv** ([ODBoolean](#)) - returns  
A flag indicating successful completion.

kODTrue	Successful completion.
kODFalse	Error occurred.

---

## DisplayHelp (OS/2) - Parameters

**sHelpFile** ([string](#)) - input  
The name of the IPF-compiled help file with an .HLP extension.

**ulPanelId** ([ODULong](#)) - input  
The ID of the help panel to be displayed.

**rv** ([ODBoolean](#)) - returns  
A flag indicating successful completion.

kODTrue	Successful completion.
kODFalse	Error occurred.

---

## DisplayHelp (OS/2) - Remarks

This method is called by a part in response to a help request. The part calls this method in response to:

- a part help menu event (HELP\_XXX)
- an [OD\\_HELP](#) window message

If an error is encountered, this method returns kODFalse and help is not displayed.

---

## DisplayHelp (OS/2) - Topics

### Class:

ODHelp

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

---

## DisplayHelpIndex (OS/2)

---

## DisplayHelpIndex (OS/2) - Syntax

The method displays the help index panel.

```
#define INCL_ODHELP
#define INCL_ODAPI
#include <os2.h>

string      sHelpFile;
ODBoolean   rv;

rv = DisplayHelpIndex(sHelpFile);
```

---

## DisplayHelpIndex (OS/2) Parameter - sHelpFile

**sHelpFile** ([string](#)) - input

The name of the IPF-compiled help file with an .HLP extension.

---

## DisplayHelpIndex (OS/2) Return Value - rv

**rv** ([ODBoolean](#)) - returns  
A flag indicating successful completion.

kODTrue	Successful completion.
kODFalse	Error occurred.

---

## DisplayHelpIndex (OS/2) - Parameters

**sHelpFile** ([string](#)) - input  
The name of the IPF-compiled help file with an .HLP extension.

**rv** ([ODBoolean](#)) - returns  
A flag indicating successful completion.

kODTrue	Successful completion.
kODFalse	Error occurred.

---

## DisplayHelpIndex (OS/2) - Remarks

This method is called by a part in response to a help index request. The part calls this method in response to a HELP\_INDEX menu event. The part should have tagged its .IPF file for indexing.

If an error is encountered, this method returns kODFalse and help is not displayed.

---

## DisplayHelpIndex (OS/2) - Topics

**Class:**  
ODHelp

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

## InitHelp (OS/2)

---

## InitHelp (OS/2) - Syntax

The method is called by the document shell to initialize help for the entire session.

```
#define INCL_ODHELP
#define INCL_ODAPI
#include <os2.h>

ODSession      *Session;

InitHelp(Session);
```

---

## InitHelp (OS/2) Parameter - Session

**Session** (ODSession \*) - input  
A reference to the current session object.

---

## InitHelp (OS/2) - Return Value

None.

---

## InitHelp (OS/2) - Parameters

**Session** (ODSession \*) - input  
A reference to the current session object.

None.

---

## InitHelp (OS/2) - Topics

**Class:**  
ODHelp

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## TerminateHelp (OS/2)

---

## TerminateHelp (OS/2) - Syntax

This method is called by the document shell to terminate help for the entire session.

```
#define INCL_ODHELP
#define INCL_ODAPI
#include <os2.h>
```

```
TerminateHelp();
```

---

## TerminateHelp (OS/2) - Return Value

None.

---

## TerminateHelp (OS/2) - Parameters

None.

---

## TerminateHelp (OS/2) - Topics

### Class:

ODHelp

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

---

## ODInfo

**Class Definition File:** INFO.IDL

### Class Hierarchy

SOMObject

ODObject

**ODInfo**

### Description

An object of the ODInfo class provides a Properties notebook, which your part uses to display standard user properties about itself or its

embedded parts.

When a document is opened, the session object creates a single information object. All parts of that document share the information object; you can obtain a reference to it by calling the session object's [GetInfo](#) method.

The information object allows you to see the Properties notebook for a part. This notebook displays properties of the part that are of interest to the user, for example, its part kind and category, size, and creation date. If you have implemented a settings extension for your part, this notebook includes a page that shows additional properties that are specific to your part.

For additional information about the Properties notebook and the settings extension, see the class description for [ODSettingsExtension](#) and the chapters on windows and menus and extending OpenDoc in the *OpenDoc Programming Guide* .

## Methods

The methods defined by the ODInfo class include:

- [ShowPartFrameInfo](#)

## Overridden Methods

There are currently no methods overridden by the ODInfo class.

---

# ShowPartFrameInfo

---

## ShowPartFrameInfo - Syntax

The method displays the Properties notebook in response to a user selection from the **Edit** menu.

```
#define INCL_ODINFO
#define INCL_ODAPI
#include <os2.h>

ODFacet      *facet;
ODBoolean    allowEditing;
ODBoolean    rv;

rv = ShowPartFrameInfo(facet, allowEditing);
```

---

## ShowPartFrameInfo Parameter - facet

**facet** (ODFacet \*) - input

A reference to the facet that indicates the window in which to display the Properties notebook. The facet's frame and part indicate which part's user properties should be displayed and edited.

---

## ShowPartFrameInfo Parameter - allowEditing

**allowEditing** (ODBoolean) - input

A flag indicating whether the part allows editing.



kODTrue	The part allows editing.
kODFalse	The part does not allow editing.

---

## ShowPartFrameInfo Return Value - rv

**rv** ([ODBoolean](#)) - returns  
A flag indicating which button the user selected in the Properties notebook.

kODTrue	The user clicked the <b>OK</b> button.
kODFalse	The user clicked the <b>Cancel</b> button.

---

## ShowPartFrameInfo - Parameters

**facet** ([ODFacet \\*](#)) - input  
A reference to the facet that indicates the window in which to display the Properties notebook. The facet's frame and part indicate which part's user properties should be displayed and edited.

**allowEditing** ([ODBoolean](#)) - input  
A flag indicating whether the part allows editing.

kODTrue	The part allows editing.
kODFalse	The part does not allow editing.

**rv** ([ODBoolean](#)) - returns  
A flag indicating which button the user selected in the Properties notebook.

kODTrue	The user clicked the <b>OK</b> button.
kODFalse	The user clicked the <b>Cancel</b> button.

---

## ShowPartFrameInfo - Remarks

You call this method in your part's [HandleEvent](#) method when the user selects the **Properties** item from the **Edit** menu and the current selection is an embedded frame border.

---

## ShowPartFrameInfo - Related Methods

### Related Methods

- [ODPart::HandleEvent](#)

---

# ShowPartFrameInfo - Topics

## Class:

ODInfo

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

## ODLink

**Class definition file:** LINK.IDL

### Class hierarchy

SOMObject

ODObject

ODRefCntObject

ODPersistentObject

ODBaseLink

**ODLink**

### Description

An object of the ODLink class represents the destinations of an OpenDoc link. Instances of this class are created and maintained by draft objects whenever the user creates a link between parts.

Linking is a mechanism for associating data in one part (the *source*) with data in another location (the *destination*) in such a way that the destination data can be updated manually or automatically whenever the source data changes. A typical example of linking allows the user to paste a spreadsheet graph into a financial report in such a way that subsequent changes to the spreadsheet are automatically reflected in the report.

Links are created during paste or drop operations. The source and destination can be in the same part, in different parts of the same document, or in different documents. A link is a persistent, one-way conduit; updating occurs from the source to the destination only. Link sources are represented by [ODLinkSource](#) objects; link destinations are represented by ODLink objects.

When the user requests a link, the following events occur:

- The destination part asks for a link object by calling its draft's [AcquireLink](#) method.
- If the source part is in the same document, the draft calls the source part's [CreateLink](#) method. The source part may refuse this request, in which case the link cannot be created; normally, however, the source part would refuse only in the event of an error.  
If the source part is in a different document, the link manager creates a cross-document link.
- If the link-source object was created successfully, the draft's [AcquireLink](#) method returns a link object to the destination part.

Every link-source object has an associated *update ID* that uniquely identifies the current generation or version of its content. The source part sets the update ID for whenever it creates or updates the content of a link-source object. The destination part stores the ID that was current when it last updated its content from the link.

The destination part reads link data from a link object's content storage unit. Because source and destination parts may attempt to access a link simultaneously, parts must acquire a lock before they can access the content storage unit. You can lock a link object by calling its [Lock](#) method; you can then obtain a reference to the link object's content storage unit by calling its [GetContentStorageUnit](#) method. You must not cache this storage unit; instead, you must call the [GetContentStorageUnit](#) method whenever you need to access the storage unit.

A destination part can register itself for automatic notification of updates by calling the [RegisterDependent](#) method of the link object. Whenever the link-source object sends updates to the link object, the link object calls the [LinkUpdated](#) method of each of its registered destination parts to notify those parts to read the new data from the link object's content storage unit.

A destination part is responsible for updating its destination content from the link object. If your part is the destination part of a link, its [LinkUpdated](#) method should read data from the link object's content storage unit and use the data to update its destination content. In addition, after your part displays the Show Link Destination Info dialog box, it should update its destination content if the user exits the dialog

box by clicking the **Update Now** button.

Each link destination tracks a single link source, but there can be several link destinations for any given link source. Within a given draft, a single link object is associated with each link-source object, even if the source is linked to multiple destinations in the draft; all destinations in the draft share this link object. For this reason, a part that maintains two or more destinations of a given link source must be careful to register only once with the corresponding link object.

A destination part's registration with a link is not permanent; the part should reregister when it is recreated from its stored data if the destination content is visible or if it could affect the layout of the part. A destination part may explicitly unregister itself from a link by calling the [UnregisterDependent](#) method of the link object. A destination part with a single destination for a particular link source should unregister when the user breaks the link at the destination, when the linked content is deleted, or when the user changes the destination from automatic to manual updating. A destination part with multiple destinations of the same link source should unregister in the same situations, provided that none of the other destinations gets updates automatically.

If your part contains a link destination, you are responsible for drawing an appropriate border around the linked content when requested to do so. If the user selects any content within the link, or checks the **Show Links** setting of the Document Info dialog box, you must show the border of the links destination whenever you draw. You can check whether to show links by calling the window's [ShouldShowLinks](#) method before drawing your part's content.

For further information on the implementation of OpenDoc links, see the descriptions of the companion classes [ODLinkSource](#) and [ODLinkSpec](#) and the chapter on data transfer in the *OpenDoc Programming Guide* .

### New methods

The methods defined for the ODLink class include:

- [GetChangeTime](#)
- [GetContentStorageUnit](#)
- [GetDescription](#)
- [GetUpdateID](#)
- [IsRegistered](#)
- [Lock](#)
- [RegisterDependent](#)
- [SetDescription](#)
- [ShowSourceContent](#)
- [Unlock](#)
- [UnregisterDependent](#)

### Overridden methods

There are currently no methods overridden by the ODLink class.

---

## GetChangeTime

---

## GetChangeTime - Syntax

This method returns the time of the last update to the link-source content.

```
#define INCL_ODLINK
#define INCL_ODAPI
#include <os2.h>

ODTime    rv;

rv = GetChangeTime();
```

---

## GetChangeTime Return Value - rv

**rv** ([ODTime](#)) - returns  
The time when the link-source content was last updated.

---

## GetChangeTime - Parameters

**rv** ([ODTime](#)) - returns  
The time when the link-source content was last updated.

---

## GetChangeTime - Topics

**Class:**  
[ODLink](#)

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## GetContentStorageUnit

---

## GetContentStorageUnit - Syntax

This method returns a reference to the storage unit containing the content of this link object.

```
#define INCL_ODLINK
#define INCL_ODAPI
#include <os2.h>

ODLinkKey      key;
ODStorageUnit  *rv;

rv = GetContentStorageUnit (key);
```

---

## GetContentStorageUnit Parameter - key

**key** ([ODLinkKey](#)) - input  
A valid link key obtained by a prior call to the [Lock](#) method.

---

# GetContentStorageUnit Return Value - rv

**rv** (ODStorageUnit \*) - returns  
A reference to this link object's content storage unit.

---

## GetContentStorageUnit - Parameters

**key** (ODLinkKey) - input  
A valid link key obtained by a prior call to the [Lock](#) method.

**rv** (ODStorageUnit \*) - returns  
A reference to this link object's content storage unit.

---

## GetContentStorageUnit - Remarks

The *key* parameter ensures thread safety. Before calling this method, you must call this link object's [Lock](#) method to obtain this key. The returned storage unit remains valid until the key is relinquished by the [Unlock](#) method.

You can read and copy but should not change the data in the returned content storage unit. The link object handles the creation and destruction of its content storage unit, so you must neither dispose of nor release the returned storage unit.

You must not cache the returned storage unit; instead, you must call this method whenever you need to access the content storage unit.

---

## GetContentStorageUnit - Exception Handling

kODErrBrokenLink	Internal error; the link source is disconnected from its destinations.
kODErrInvalidLinkKey	The <i>key</i> parameter is not a valid key for this link.

---

## GetContentStorageUnit - Topics

**Class:**  
ODLink

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)

---

# GetDescription (OS/2)

---

## GetDescription (OS/2) - Syntax

This method is called by the destination part to get the link's description.

```
#define INCL_ODLINK
#define INCL_ODAPI
#include <os2.h>

ODLinkDescription *Desc;

GetDescription(Desc);
```

---

## GetDescription (OS/2) Parameter - Desc

**Desc** ([ODLinkDescription \\*](#)) - output  
A string containing a comment describing part-specific information.

---

## GetDescription (OS/2) - Return Value

None.

---

## GetDescription (OS/2) - Parameters

**Desc** ([ODLinkDescription \\*](#)) - output  
A string containing a comment describing part-specific information.

None.

---

## GetDescription (OS/2) - Remarks

The part description could include information such as the part kind, name, or type, or more specific information as to how the linked content is being used in the destination part.

---

# GetDescription (OS/2) - Topics

## Class:

ODLink

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

---

## GetUpdateID

---

## GetUpdateID - Syntax

This method returns the update ID that uniquely identifies the current generation or version of link-source content.

```
#define INCL_ODLINK
#define INCL_ODAPI
#include <os2.h>
```

```
ODUpdateID    rv;
```

```
rv = GetUpdateID();
```

---

## GetUpdateID Return Value - rv

**rv** ([ODUpdateID](#)) - returns

The current update ID of the link-source content.

---

## GetUpdateID - Parameters

**rv** ([ODUpdateID](#)) - returns

The current update ID of the link-source content.

---

## GetUpdateID - Remarks

If your part contains a destination of this link, you can call this method to determine the current version of the content of the link-source object. You can compare the returned update ID with a previously saved ID to establish whether the source content has been modified since you last read it. There is no implicit ordering of update ID values; they offer tests for equality only, with no indication of the time or nature of any link changes.

You can arrange for your part to receive automatic notification of content updates by calling the [RegisterDependent](#) method.

---

## GetUpdateID - Exception Handling

kODErrBrokenLink

Internal error; the link-source object is disconnected from its destination.

---

## GetUpdateID - Related Methods

### Related Methods

- [ODLink::RegisterDependent](#)
- [ODLinkSource::ContentUpdated](#)

---

## GetUpdateID - Topics

### Class:

ODLink

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## IsRegistered (OS/2)

---

## IsRegistered (OS/2) - Syntax

This method indicates whether the part is registered with the link object using the [RegisterDependent](#) method.

```
#define INCL_ODLINK
#define INCL_ODAPI
#include <os2.h>
```



```
ODBoolean rv;  
  
rv = IsRegistered();
```

---

## IsRegistered (OS/2) Return Value - rv

**rv** ([ODBoolean](#)) - returns  
A flag indicating whether the part is registered with the link object.

kODTrue	The part is registered with the link object.
kODFalse	The part is not registered with the link object.

---

## IsRegistered (OS/2) - Parameters

**rv** ([ODBoolean](#)) - returns  
A flag indicating whether the part is registered with the link object.

kODTrue	The part is registered with the link object.
kODFalse	The part is not registered with the link object.

---

## IsRegistered (OS/2) - Topics

**Class:**  
ODLink

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## Lock

---

## Lock - Syntax

This method locks this link object, ensuring exclusive, read-only access to its content storage unit.

```
#define INCL_ODLINK
#define INCL_ODAPI
#include <os2.h>

ODULong      wait;
ODLinkKey    *key;
ODBoolean    rv;

rv = Lock(wait, key);
```

---

## Lock Parameter - wait

**wait** (ODULong) - input  
The time interval to wait for access to be granted.

---

## Lock Parameter - key

**key** (ODLinkKey \*) - output  
A valid link key. If access is not granted, this value is an undefined, invalid key.

---

## Lock Return Value - rv

**rv** (ODBoolean) - returns  
A flag indicating whether access is granted.

kODTrue	Access is granted.
kODFalse	Access is denied.

---

## Lock - Parameters

**wait** (ODULong) - input  
The time interval to wait for access to be granted.

**key** (ODLinkKey \*) - output  
A valid link key. If access is not granted, this value is an undefined, invalid key.

**rv** (ODBoolean) - returns  
A flag indicating whether access is granted.

kODTrue	Access is granted.
kODFalse	Access is denied.

---

## Lock - Remarks

To ensure thread-safe access, you must call this method to acquire a valid link key before you can read the link data. This method grants read-only access; a destination part cannot modify a link's content.

A link may be locked by only one object at a time; nested calls to the Lock method deny access.

While your part has the link locked, you must pass the key returned in the *key* output parameter to all methods that access the link. When you are finished using the link, you must pass this key to the [Unlock](#) method to unlock the link.

---

## Lock - Exception Handling

kODErrBrokenLink

Internal error; the link-source object is disconnected from its destination.

---

## Lock - Related Methods

### Related Methods

- [ODLink::GetContentStorageUnit](#)
  - [ODLink::Unlock](#)
- 

## Lock - Topics

### Class:

ODLink

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## RegisterDependent

---

## RegisterDependent - Syntax

This method puts the specified destination part on the list of parts to be notified whenever the link is updated.

```
#define INCL_ODLINK
#define INCL_ODAPI
#include <os2.h>

ODPart      *clientPart;
ODUpdateID   id;

RegisterDependent(clientPart, id);
```

-----

## RegisterDependent Parameter - clientPart

**clientPart** (ODPart \*) - input

A reference to the destination part wishing to be notified of changes to the content of the link source.

-----

## RegisterDependent Parameter - id

**id** (ODUpdateID) - input

The update ID of the content the specified part last read from the link source. This parameter can also be set to the following value:

kODUnknownUpdate

The part does not keep track of link change identifications. Setting this value ensures that your part's [LinkUpdated](#) method is called.

-----

## RegisterDependent - Return Value

None.

-----

## RegisterDependent - Parameters

**clientPart** (ODPart \*) - input

A reference to the destination part wishing to be notified of changes to the content of the link source.

**id** (ODUpdateID) - input

The update ID of the content the specified part last read from the link source. This parameter can also be set to the following value:

kODUnknownUpdate

The part does not keep track of link change identifications. Setting this value ensures that your part's [LinkUpdated](#) method is called.

None.

-----

# RegisterDependent - Remarks

If the your part is a destination of this link, you can call this method to register your part as a dependent of this link object. Dependent parts receive automatic notification of any changes to this link's content. You can remove your part from the list of dependents by calling the [UnregisterDependent](#) method.

If the link's current update ID differs from the *id* parameter, OpenDoc immediately calls the specified client part's [LinkUpdated](#) method. You should pass the constant `KODUnknownUpdate` as the value of the *id* parameter when the link is first created; doing so ensures that your part's [LinkUpdated](#) method is called.

**Note:** You should not call this method if your part is already a registered dependent of this link object, for example, because your part contains another destination of link source corresponding to this link object.

---

# RegisterDependent - Exception Handling

`kODErrBrokenLink`

Internal error; the link source is disconnected from its destination.

---

# RegisterDependent - Related Methods

## Related Methods

- [ODLink::GetUpdateID](#)
  - [ODLink::UnregisterDependent](#)
  - [ODPart::LinkUpdated](#)
- 

# RegisterDependent - Topics

## Class:

`ODLink`

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

# SetDescription (OS/2)

---

# SetDescription (OS/2) - Syntax

This method is called by the destination part to set the link's description.

```
#define INCL_ODLINK
#define INCL_ODAPI
#include <os2.h>

ODLinkDescription    *Desc;

SetDescription(Desc);
```

---

## SetDescription (OS/2) Parameter - Desc

**Desc** (ODLinkDescription \*) - input  
A string containing a comment describing part-specific information which the write of the part containing the link would like to publish.

---

## SetDescription (OS/2) - Return Value

None.

---

## SetDescription (OS/2) - Parameters

**Desc** (ODLinkDescription \*) - input  
A string containing a comment describing part-specific information which the write of the part containing the link would like to publish.

None.

---

## SetDescription (OS/2) - Remarks

The part description could include information such as the part kind, name, or type, or more specific information as to how the linked content is being used in the destination part. This method can be called multiple times after the link has been established, and each time, the new comment replaces the old one.

---

## SetDescription (OS/2) - Topics

**Class:**  
ODLink

Select an item:

---

## ShowSourceContent

---

### ShowSourceContent - Syntax

This method requestes the source part of the link to make the source content visible.

```
#define INCL_ODLINK  
#define INCL_ODAPI  
#include <os2.h>
```

```
ShowSourceContent();
```

---

### ShowSourceContent - Return Value

None.

---

### ShowSourceContent - Parameters

None.

---

### ShowSourceContent - Remarks

This method causes the source part's [RevealLink](#) method to be called, making the source content visible.

If this method throws an exception, you should display an alert message informing the user that the link-source content could not be shown.

---

### ShowSourceContent - Exception Handling

kODErrBrokenLink

Internal error; the link-source object is disconnected from its destinations.

This method may also throw exceptions that were raised by the part's [RevealLink](#) method.

---

## ShowSourceContent - Related Methods

### Related Methods

- [ODPart::RevealLink](#)
- [ODPart::EditInLinkAttempted](#)

---

## ShowSourceContent - Topics

### Class:

[ODLink](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## Unlock

---

## Unlock - Syntax

This method unlocks this link object, relinquishing access to its content storage unit.

```
#define INCL_ODLINK  
#define INCL_ODAPI  
#include <os2.h>
```

```
ODLinkKey    key;
```

```
Unlock(key);
```

---

## Unlock Parameter - key



**key** ([ODLinkKey](#)) - input  
A valid link key obtained by a prior call to the [Lock](#) method.

---

## Unlock - Return Value

None.

---

## Unlock - Parameters

**key** ([ODLinkKey](#)) - input  
A valid link key obtained by a prior call to the [Lock](#) method.

None.

---

## Unlock - Remarks

You should call this method as soon as possible after accessing the content storage unit of this link object.

The *key* parameter should be a valid key obtained by an earlier call to the [Lock](#) method. After this method executes successfully, the specified key is no longer valid and access to the link's content is relinquished.

---

## Unlock - Exception Handling

kODErrBrokenLink

Internal error; the link-source object is disconnected from its destinations.

kODErrInvalidLinkKey

The *key* parameter is not a valid key for this link.

---

## Unlock - Related Methods

### Related Methods

- [ODLink::Lock](#)
- 

## Unlock - Topics

**Class:**

ODLink

Select an item:

[Syntax](#)[Parameters](#)[Returns](#)[Remarks](#)[Exception Handling](#)[Related Methods](#)

---

## UnregisterDependent

---

### UnregisterDependent - Syntax

This method removes the previously registered destination part from the list of parts to be notified whenever the link is updated.

```
#define INCL_ODLINK
#define INCL_ODAPI
#include <os2.h>

ODPart      *clientPart;

UnregisterDependent(clientPart);
```

---

### UnregisterDependent Parameter - clientPart

**clientPart** (ODPart \*) - input

A reference to a destination part to be unregistered.

---

### UnregisterDependent - Return Value

None.

---

### UnregisterDependent - Parameters

**clientPart** (ODPart \*) - input

A reference to a destination part to be unregistered.

None.

---

## UnregisterDependent - Remarks

If your part is a destination of this link, you can call this method to unregister your part as a dependent. Dependent parts receive automatic notification of any changes to the link's content. You can reregister your part by calling the [RegisterDependent](#) method.

The UnregisterDependent method returns without error if the specified destination part was not previously registered as a dependent of this link.

---

## UnregisterDependent - Related Methods

### Related Methods

- [ODLink::RegisterDependent](#)
- 

## UnregisterDependent - Topics

### Class:

ODLink

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## ODLinkSource

**Class definition file:** LINKSRC.IDL

### Class hierarchy

```
SOMObject
  ODOObject
    ODRefCntObject
      ODPersistentObject
        ODBaseLinkSource
          ODLinkSource
```

### Description

An object of the ODLinkSource class represents the source of an OpenDoc link. Link-source objects are created and maintained by draft objects whenever the user creates a link between parts.

Linking is a mechanism for associating data in one part (the *source* ) with data in another location (the *destination* ) in such a way that the destination data can be updated either manually or automatically whenever the source data changes. A typical example of linking allows the user to paste a spreadsheet graph to a financial report in such a way that subsequent changes to the spreadsheet are automatically reflected

in the report.

Links are created during paste or drop operations. The source and destination can be in the same part, in different parts of the same document, or in different documents. A link is a persistent, one-way conduit; updating occurs from the source to the destination only. Link sources are represented by `ODLinkSource` objects; link destinations are represented by `ODLink` objects. Source parts reference only link-source objects, and destination parts reference only link objects. These reference restrictions are important in ensuring the proper behavior when linked content is copied or moved.

When a source part that supports linking writes to the clipboard or drag-and-drop object, it uses a link specification to advertise its ability to create links. The source part creates a link specification (an object of class `ODLinkSpec`) by calling its draft's `CreateLinkSpec` method. In addition to writing the source content to the clipboard or drag-and-drop object, the source part writes link-specification data to the `kODPropLinkSpec` property. Subsequently, the user who pastes or drops the source data to a destination part can request a link by means of a Paste As dialog box.

When the user requests a link, the destination part asks for a link object by calling its draft's `AcquireLink` method. If the source part is in the same document, the draft calls the source part's `CreateLink` method. If the source part has not already created a link-source object for the source content described by the link specification, the source part calls its draft's `CreateLinkSource` method to create a link-source object. The source part's draft is responsible for both the transitory and persistent storage of its link-source object. If the source part is in a different document, the link manager creates a cross-document link.

When a source part is recreated from its stored data, its `InitPartFromStorage` method can call its draft object's `AcquireLinkSource` method to recreate the link-source object from storage.

If your part contains a link source, you are responsible for drawing an appropriate border around the linked content when requested to do so. If the user selects any content within the link, or checks the **Shows Links** setting of the Document Info dialog box, you must show the border of the link source whenever you draw. You can check whether to show links by calling the window's `ShouldShowLinks` method before drawing your part's content. If the user cuts content from your part that includes a link source, your part relinquishes ownership of the link-source object. If the user subsequently undoes the action, you must call the `SetSourcePart` method of the link-source objects to reclaim ownership.

Every link source has an associated *update ID* that uniquely identifies the current generation or version of its content. The source part sets the update ID for when ever it creates or updates the content of a link-source object. The destination part stores the ID that was current when it last updated its content from the link.

For further information on the implementation of OpenDoc links, see the class descriptions for `ODLink` and `ODLinkSpec`, and the chapter on data transfer in the *OpenDoc Programming Guide*.

### Creating Multiple Destinations for a Link

If your part is the source part for a link, its `CreateLink` method may be called multiple times with the same link specification to create multiple destinations for the same link. If you write data to the content storage unit of the link-source object when you first create it, your `CreateLink` method can simply return the link-source object on subsequent calls; however, if you write promises, you need to ensure that all part kinds originally promised are present in the content storage unit. The initial destination caused your part to fulfill promises of the part kinds that it copied but might have left unfulfilled promises for other part kinds. If the draft was saved, those unfulfilled promises were removed. To add a new destination, you need to clear the content storage unit and rewrite promises for all part kinds. First, call the link-source object's `GetUpdateID` method to get its current update ID. Next, pass that ID as the parameter when you call the `Clear` method. Because the ID is not being changed, the link-source object keeps track of the nonpromise part kinds that were in the content storage unit when you cleared it. When you subsequently write promises for those part kinds, the link-source object automatically forces the source part to fulfill them. Thus, you can ensure that the content storage unit contains all the data needed by the initial destination part and promises for the other part kinds that might be needed by the new destination part.

### Data Transfer from Source to Destination

The process of transferring content from a link source to a link destination requires three steps. First, the source part transfers content to the link-source object. Second, the link-source object transfers content to each of its associated link objects. Third, each destination part must read the content from its link object.

#### Update Mode

Every link-source object has an update mode that affects the first two steps of the transfer from source part to destination part. The source part should use the update mode to determine when to transfer content to the link-source object. In addition, the link-source object uses the update mode to determine when to send updated content to its associated link object in other documents. (Regardless of the update mode, whenever the source part updates the link-source object, the link-source object immediately sends the updated content to its associated link objects in the same document.)

- The *automatic* update mode indicates that the source part should update the link-source object whenever the source content has changed; the link-source object sends those updates to destinations in other documents whenever its draft is saved. This mode is the default.
- The *manual* update mode indicates that the source part should update the link-source object only when the user clicks the **Update Now** button in the Link Source Info dialog box; the link-source object immediately sends those updates to destinations in other documents.

#### Source Part to Link-Source Object

If your part is the source part of a link, you should use the update mode of the link-source object to determine when to transfer data. Several methods are required to effect the data transfer.

Because source and destination parts may attempt to access a link simultaneously, parts must acquire a lock before they transfer link data. You can lock the link-source object by calling its [Lock](#) method.

Once the link-source object is locked, you can obtain a reference to its content storage unit by calling the [GetContentStorageUnit](#) method. You must not cache that storage unit; instead, you must call the [GetContentStorageUnit](#) method whenever you need to access the storage unit.

If you are updating the content, as opposed to writing initial content, you should call the link-source object's [Clear](#) method before you write to the content storage unit.

You can call the appropriate methods of the content storage unit to write data to it. If you need to clone objects, call the cloning methods of your part's draft object. For more information, see the descriptions of the classes [ODStorageUnit](#) and [ODDraft](#).

After writing initial content or updates to the link-source object's content storage unit, you must notify the link-source object that you have updated its content by calling its [ContentUpdated](#) method. Finally, you should unlock the link-source object by calling its [Unlock](#) method.

#### *Link-Source Object to Link Object*

The link-source object automatically transfers new content to its associated link objects. This step requires no action by either the source part or the destination part. As mentioned previously, the time when this transfer occurs varies depending on the update mode of the link-source object and whether the link object is in the same document as the link-source object.

#### *Link Object to Destination Part*

Every destination part is responsible for updating its destination content from the link object. A destination part can register itself for automatic notification of updates by calling the [RegisterDependent](#) method of a link object. A registered destination part should update its content from the link content when it is notified of an update; an unregistered part should update its content when the user clicks the **Update Now** button in the Show Link Destination Info dialog box.

### **New methods**

The methods defined by the ODLinkSource class include:

- [Clear](#)
- [ContentUpdated](#)
- [GetChangeTime](#)
- [GetContentStorageUnit](#)
- [GetDescription](#)
- [GetUpdateID](#)
- [IsAutoUpdate](#)
- [Lock](#)
- [SetAutoUpdate](#)
- [SetDescription](#)
- [SetSourcePart](#)
- [Unlock](#)

### **Overridden Methods**

There are currently no methods overridden by the ODLinkSource class.

## Clear

## Clear - Syntax

This method clears the content of this link-source object.

```
#define INCL_ODLINKSOURCE
#define INCL_ODAPI
#include <os2.h>

ODUpdateID    id;
ODLinkKey     key;
```

```
Clear(id, key);
```

---

## Clear Parameter - id

**id** ([ODUpdateID](#)) - input

The update ID associated with the new empty content.

---

## Clear Parameter - key

**key** ([ODLinkKey](#)) - input

A valid link key obtained by a prior call to the [Lock](#) method.

---

## Clear - Return Value

None.

---

## Clear - Parameters

**id** ([ODUpdateID](#)) - input

The update ID associated with the new empty content.

**key** ([ODLinkKey](#)) - input

A valid link key obtained by a prior call to the [Lock](#) method.

None.

---

## Clear - Remarks

If your part is the source part for this link-source object, you call this method to remove the link-source object's previous content before updating the source data. This method removes all data in the content property (kODPropContents) from the link-source object's content storage unit.

The *id* parameter identifies the new version of this link-source object's content; its value depends on the circumstances in which this method is called:

- If your part originated the changes to its source content, the *id* parameter should be a new update ID that you obtained by calling the session object's [UniqueUpdateID](#) method.

- If the changes occurred because the source content of this link contains the destination content of another link that was updated, the *id* parameter should be the updated ID that your part received when it was notified to update the destination content within the source content for this link.
- If your part originally created this link-source object with promise data and its [CreateLink](#) method is being called again to create a new link destination, the *id* parameter should be the current update ID for this link-source object. You can call the [GetUpdateID](#) method to get the current update ID.

To add a new destination for this link-source object, you need to clear the content storage unit and rewrite promises for all part kinds. Because the ID is not being changed, the link-source object keeps track of the nonpromise part kinds that were in the content storage unit when you cleared it. When you subsequently rewrite promises for those part kinds, the link-source object automatically forces the source part to fulfill them; you should *not* call the link-source object's [ContentUpdated](#) method after rewriting those promises.

The *key* parameter ensures thread safety. Before calling this method, you must call this link-source object's [Lock](#) method to obtain this key.

-----

## Clear - Exception Handling

kODErrInvalidLinkKey

The *key* parameter is not a valid key for this link-source object.

kODErrInvalidPermissions

Draft permissions do not allow modifications.

kODErrUnknownUpdateID

The specified update ID is the reserved value kODUnknownUpdate.

-----

## Clear - Related Methods

### Related Methods

- [ODLinkSource::GetContentStorageUnit](#)
- [ODLinkSource::GetUpdateID](#)
- [ODLinkSource::Lock](#)
- [ODSession::UniqueUpdateID](#)

-----

## Clear - Topics

### Class:

[ODLinkSource](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

-----

## ContentUpdated

---

# ContentUpdated - Syntax

This method notifies this link-source object that its content has been updated.

```
#define INCL_ODLINKSOURCE
#define INCL_ODAPI
#include <os2.h>

ODUpdateID    id;
ODLinkKey     key;

ContentUpdated(id, key);
```

---

## ContentUpdated Parameter - id

**id** ([ODUpdateID](#)) - input  
The update ID associated with the new content.

---

## ContentUpdated Parameter - key

**key** ([ODLinkKey](#)) - input  
A valid link key obtained by a prior call to the [Lock](#) method.

---

## ContentUpdated - Return Value

None.

---

## ContentUpdated - Parameters

**id** ([ODUpdateID](#)) - input  
The update ID associated with the new content.

**key** ([ODLinkKey](#)) - input  
A valid link key obtained by a prior call to the [Lock](#) method.

None.

---



# ContentUpdated - Remarks

If your part is the source part for this link-source object, you call this method after writing the initial content to the content storage unit of this link-source object and after making changes to the content of that storage unit.

The *id* parameter identifies the new version of the link source object's content; its value depends on the circumstances in which this method is called:

- If you called the [Clear](#) method before calling this method, the *id* parameter should be the same update ID that you passed to the [Clear](#) method.
- If your part originated the changes to its source content, the *id* parameter should be a new update ID that you can be obtained by calling the session object's [UniqueUpdateID](#) method.
- If the changes occurred because the source content of this link contains the destination content of another link that was updated, the *id* parameter should be the updated ID that your part received when it was notified to update the destination content within the source content for this link.

The *key* parameter ensures thread safety. Before calling this method, you must call the [Lock](#) method to obtain this key.

---

# ContentUpdated - Exception Handling

kODErrInvalidLinkKey

The *key* parameter is not a valid key for this link-source object.

kODErrUnknownUpdateID

The specified update ID is the reserved value kODUnknownUpdate.

---

# ContentUpdated - Related Methods

## Related Methods

- [ODFrame::ChangeLinkStatus](#)
- [ODFrame::ContentUpdated](#)
- [ODLinkSource::GetUpdateID](#)
- [ODLinkSource::Lock](#)
- [ODPart::EmbeddedFrameUpdated](#)
- [ODSession::UniqueUpdateID](#)

---

# ContentUpdated - Topics

## Class:

[ODLinkSource](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

# GetChangeTime

---

## GetChangeTime - Syntax

This method returns the time that this link-source object was last updated.

```
#define INCL_ODLINKSOURCE
#define INCL_ODAPI
#include <os2.h>

ODTime    rv;

rv = GetChangeTime();
```

---

## GetChangeTime Return Value - rv

**rv** ([ODTime](#)) - returns  
The time when this link source was last updated.

---

## GetChangeTime - Parameters

**rv** ([ODTime](#)) - returns  
The time when this link source was last updated.

---

## GetChangeTime - Topics

**Class:**  
ODLinkSource

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## GetContentStorageUnit

---

# GetContentStorageUnit - Syntax

This method returns a reference to the content storage unit for this link-source object.

```
#define INCL_ODLINKSOURCE
#define INCL_ODAPI
#include <os2.h>

ODLinkKey      key;
ODStorageUnit  *rv;

rv = GetContentStorageUnit(key);
```

---

## GetContentStorageUnit Parameter - key

**key** ([ODLinkKey](#)) - input  
A valid link key obtained by a prior call to the [Lock](#) method.

---

## GetContentStorageUnit Return Value - rv

**rv** ([ODStorageUnit](#) \*) - returns  
A reference to this link-source object's content storage unit.

---

## GetContentStorageUnit - Parameters

**key** ([ODLinkKey](#)) - input  
A valid link key obtained by a prior call to the [Lock](#) method.

**rv** ([ODStorageUnit](#) \*) - returns  
A reference to this link-source object's content storage unit.

---

## GetContentStorageUnit - Remarks

The *key* parameter ensures thread safety. Before calling this method, you must call the [Lock](#) method to obtain this key.

The returned storage unit remains valid until the key is relinquished by the [Unlock](#) method. The link-source object handles the creation and destruction of its content storage unit, so you must neither dispose of nor release the returned storage unit.

You must not cache the returned storage unit; instead, you must call this method whenever you need to access the content storage unit.

---

# GetContentStorageUnit - Exception Handling

kODErrInvalidLinkKey

The `key` parameter is not a valid key for this link-source object.

## GetContentStorageUnit - Related Methods

### Related Methods

- [ODLinkSource::Lock](#)
- [ODLinkSource::Unlock](#)

## GetContentStorageUnit - Topics

### Class:

[ODLinkSource](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

## GetDescription

## GetDescription - Syntax

This method is called by the source part to get the link-source objects's description.

```
#define INCL_ODLINKSOURCE
#define INCL_ODAPI
#include <os2.h>

ODLinkDescription    *desc;

GetDescription(desc);
```

## GetDescription Parameter - desc

**desc** ([ODLinkDescription \\*](#)) - output  
A string containing part-specific information.

---

## GetDescription - Return Value

None.

---

## GetDescription - Parameters

**desc** ([ODLinkDescription \\*](#)) - output  
A string containing part-specific information.

None.

---

## GetDescription - Remarks

The part description could include information such as the part kind, name, and type, and more specific information as to how the linked content is being used in the destination part.

---

## GetDescription - Topics

**Class:**  
ODLink

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

## GetUpdateID

---

## GetUpdateID - Syntax

This method returns the update ID that uniquely identifies the current generation or version of this link-source object's content.

```
#define INCL_ODLINKSOURCE
#define INCL_ODAPI
#include <os2.h>

ODUpdateID    rv;

rv = GetUpdateID();
```

---

## GetUpdateID Return Value - rv

**rv** ([ODUpdateID](#)) - returns  
The update ID of this link-source object's content.

---

## GetUpdateID - Parameters

**rv** ([ODUpdateID](#)) - returns  
The update ID of this link-source object's content.

---

## GetUpdateID - Remarks

If your part is the source part for this link-source object, you can call this method to determine the current version of this link-source object's content.

You can compare the returned update ID with a previously saved ID to establish whether the content needs to be updated. There is no implicit ordering of update ID values; they offer tests for equality only, with no indication of the time or nature of any link changes.

---

## GetUpdateID - Related Methods

### Related Methods

- [ODLinkSource::GetChangeTime](#)

---

## GetUpdateID - Topics

**Class:**  
ODLinkSource

Select an item:

---

# IsAutoUpdate

---

## IsAutoUpdate - Syntax

This method indicates whether the link-source object' update mode is automatic.

```
#define INCL_ODLINKSOURCE
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    rv;

rv = IsAutoUpdate();
```

---

## IsAutoUpdate Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the update mode for this link-source object is automatic.

kODTrue

Update mode is set to is automatic. This is the default setting.

kODFalse

Update mode is set to manual.

---

## IsAutoUpdate - Parameters

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the update mode for this link-source object is automatic.

kODTrue

Update mode is set to is automatic. This is the default setting.

kODFalse

Update mode is set to manual.

---

## IsAutoUpdate - Remarks

If your part is the source part of this link-source object, you can call this method to determine when your part should update the content of this link-source object.

If this method returns `kODTrue`, you should update this link-source object whenever the source content is changed. If it returns `kODFalse`, your part should update this link-source object when the user clicks the **Update Now** button in the Link Source Info dialog box. (Whenever your part updates this link-source object, it should also call the [ContentUpdated](#) method.)

The update mode can be changed by the [SetAutoUpdate](#) method.

---

## IsAutoUpdate - Related Methods

### Related Methods

- [ODLinkSource::ContentUpdated](#)
- [ODLinkSource::SetAutoUpdate](#)

---

## IsAutoUpdate - Topics

### Class:

`ODLinkSource`

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## Lock

---

## Lock - Syntax

This method locks this link-source object, ensuring exclusive access to its content storage unit.

```
#define INCL_ODLINKSOURCE
#define INCL_ODAPI
#include <os2.h>

ODULong      wait;
ODLinkKey    *key;
ODBoolean    rv;

rv = Lock(wait, key);
```

---

## Lock Parameter - wait



**wait** (ODULong) - input  
The time interval to wait for access to be granted or 0 if the link-source object is to be lock immediately.

---

## Lock Parameter - key

**key** (ODLinkKey \*) - output  
A valid link key. If access is denied, this parameter contains an undefined, invalid key.

---

## Lock Return Value - rv

**rv** (ODBoolean) - returns  
A flag indicating whether access is granted.

kODTrue	Access is granted.
kODFalse	Access is denied.

---

## Lock - Parameters

**wait** (ODULong) - input  
The time interval to wait for access to be granted or 0 if the link-source object is to be lock immediately.

**key** (ODLinkKey \*) - output  
A valid link key. If access is denied, this parameter contains an undefined, invalid key.

**rv** (ODBoolean) - returns  
A flag indicating whether access is granted.

kODTrue	Access is granted.
kODFalse	Access is denied.

---

## Lock - Remarks

To ensure thread-safe access, you must call this method to acquire a valid link key before you write the link data. This method grants exclusive access to this link-source object's content; nested calls to the Lock method deny access.

While your part has the link-source object locked, you must pass the key returned in the *key* output parameter to all methods that access or modify the link-source object, such as the [Clear](#), [ContentUpdated](#), and [GetContentStorageUnit](#) methods. When you are finished modifying the link-source object, you must pass this key to the [Unlock](#) method to unlock the link-source object.

---

# Lock - Exception Handling

kODErrBrokenLink

Internal error; the link-source object disconnected from its destination.

## Lock - Related Methods

### Related Methods

- [ODLinkSource::Clear](#)
- [ODLinkSource::ContentUpdated](#)
- [ODLinkSource::GetContentStorageUnit](#)
- [ODLinkSource::Unlock](#)

## Lock - Topics

### Class:

[ODLinkSource](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

## SetAutoUpdate

## SetAutoUpdate - Syntax

This method sets this link-source object's update mode to automatic or manual.

```
#define INCL_ODLINKSOURCE
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    automatic;
```

```
SetAutoUpdate (automatic);
```

## SetAutoUpdate Parameter - automatic

**automatic** (ODBoolean) - input

A flag indicating whether the update mode for this link-source object is to be automatic.

kODTrue

The update mode is to be automatic.

kODFalse

The update mode is to be manual.

-----

## SetAutoUpdate - Return Value

None.

-----

## SetAutoUpdate - Parameters

**automatic** (ODBoolean) - input

A flag indicating whether the update mode for this link-source object is to be automatic.

kODTrue

The update mode is to be automatic.

kODFalse

The update mode is to be manual.

None.

-----

## SetAutoUpdate - Remarks

Your part should use the new update mode to determine when to update the content of this link-source object. If the *automatic* parameter is kODTrue, your part should update this link-source object whenever the source content is changed. If the parameter is kODFalse, your part should update this link-source object when the user clicks the **Update Now** button in the Link Source Info dialog box. (Whenever your part updates this link-source object, it should also call the [ContentUpdated](#) method.)

-----

## SetAutoUpdate - Related Methods

### Related Methods

- [ODLinkSource::ContentUpdated](#)
- [ODLinkSource::IsAutoUpdate](#)

-----

## SetAutoUpdate - Topics

**Class:**

ODLinkSource

Select an item:

[Syntax](#)[Parameters](#)[Returns](#)[Remarks](#)[Related Methods](#)

---

## SetDescription

---

### SetDescription - Syntax

This method is called by the source part to set the link source's description.

```
#define INCL_ODLINKSOURCE
#define INCL_ODAPI
#include <os2.h>

ODLinkDescription    *desc;

SetDescription(desc);
```

---

### SetDescription Parameter - desc

**desc** ([ODLinkDescription \\*](#)) - input  
A string containing part-specific information to be published.

---

### SetDescription - Return Value

None.

---

### SetDescription - Parameters

**desc** ([ODLinkDescription \\*](#)) - input  
A string containing part-specific information to be published.

None.

---

## SetDescription - Remarks

The part description could include information such as the part kind, name, and type, and more specific information as to how the linked content is being used in the destination part. This method can be called multiple times after the link has been established. With each call, the new comment replaces the old one.

---

## SetDescription - Topics

### Class:

ODLink

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

---

## SetSourcePart

---

## SetSourcePart - Syntax

This method establishes the source part that is to maintain this link-source object.

```
#define INCL_ODLINKSOURCE
#define INCL_ODAPI
#include <os2.h>

ODStorageUnit      *sourcePartSU;

SetSourcePart (sourcePartSU);
```

---

## SetSourcePart Parameter - sourcePartSU

**sourcePartSU** (ODStorageUnit \*) - input

A reference to the storage unit for the new source part or kODNULL if the link source no longer references the part's storage unit (which can be undone by calling the part's storage unit).

---

# SetSourcePart - Return Value

None.

---

# SetSourcePart - Parameters

**sourcePartSU** (ODStorageUnit \*) - input

A reference to the storage unit for the new source part or KODNULL if the link source no longer references the part's storage unit (which can be undone by calling the part's storage unit).

None.

---

# SetSourcePart - Remarks

Both parts and container applications call this method, typically when this link-source object is cloned during data-transfer operations involving linked content.

If the user cuts content from your part that includes a link source, your part relinquishes ownership of the link-source object. If the user subsequently undoes that action, you must call this method to reclaim ownership.

After this method executes successfully, the reference count of the storage unit of the previous source part is decremented and the reference count of the new source part's storage unit is incremented.

---

# SetSourcePart - Topics

## Class:

ODLinkSource

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

---

# Unlock

---

# Unlock - Syntax

This method unlocks this link-source object, relinquishing access to its content storage unit.

```
#define INCL_ODLINKSOURCE
#define INCL_ODAPI
#include <os2.h>

ODLinkKey    key;

Unlock(key);
```

-----

## Unlock Parameter - key

**key** ([ODLinkKey](#)) - input  
A valid key obtained by a prior call to the [Lock](#) method.

-----

## Unlock - Return Value

None.

-----

## Unlock - Parameters

**key** ([ODLinkKey](#)) - input  
A valid key obtained by a prior call to the [Lock](#) method.

None.

-----

## Unlock - Remarks

You should call this method as soon as possible after accessing or modifying the content storage unit of this link-source object.

The *key* parameter should be a valid key obtained by an earlier call to the [Lock](#) method. After this method executes successfully, the specified key is no longer valid and access to the link-source object's content is relinquished.

-----

## Unlock - Exception Handling

kODErrInvalidLinkKey

The *key* parameter is not a valid key for this link-source object.

---

# Unlock - Related Methods

## Related Methods

- [ODLinkSource::Lock](#)
- 

# Unlock - Topics

## Class:

ODLinkSource

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

---

# ODLinkSpec

**Class definition file:** LINKSPEC.IDL

## Class hierarchy

SOMObject  
  ODObject  
    ODBaseLinkSpec  
      **ODLinkSpec**

## Description

An object of the ODLinkSpec class advertises a part's ability to create a link to the data it is transferring.

A link specification is a signal that a link can be made from content in a specified source part. The link specification remains valid as long as its source part exists and the document containing it remains opened. After the document is closed, the link specification becomes meaningless.

When a source part that supports linking writes to the clipboard or drag-and-drop object, it uses a link specification to advertise its ability to create links. The source part creates a link specification by calling its draft's [CreateLinkSpec](#) method, passing as parameters a reference to itself and data sufficient to enable the source part to identify the selected content. The data in a link specification is private to the source part; it is returned to the source part if the part's [CreateLink](#) method is called to create a link. Because the link specification is valid only during the lifetime of its source part, the data can contain pointers to information maintained by the part.

A link specification provides methods for writing or reading itself to and from a focused storage unit. In addition to writing content to the storage unit of the data-transfer object (clipboard or drag-and-drop object), the source part calls the link specification's [WriteLinkSpec](#) method to write the link specification to the kODPropLinkSpec property.

A user who pastes or drops the source data to a destination part can request a link by means of a Paste As dialog box. The destination part creates an empty link specification by calling the draft's [CreateLinkSpec](#) method, passing null for the source part and data parameters. The destination part calls the [ReadLinkSpec](#) method of the empty link specification to read from the kODPropLinkSpec property of the content storage unit. The destination part then creates the link by passing the link specification to its draft's [AcquireLink](#) method, which in turn passes the link specification to the source part's [CreateLink](#) method.

Whenever the source part writes a link specification to the clipboard, it should save the clipboard's current update ID, as returned by the clipboard's [GetUpdateID](#) method. The part must remove the link specification from the clipboard when either of the following conditions is true:

- It becomes infeasible to create the link, for example, because the potential source content is deleted or modified.



- The part's [Release](#) method is called.

The source part can check the clipboard update ID and compare it with the save ID to determine whether its content is still on the clipboard.

For further information on the clipboard and drag-and-drop object, see the descriptions of the classes [ODClipboard](#) and [ODDragAndDrop](#). For further information on the implementation of OpenDoc links, see the descriptions for the classes [ODLink](#) and [ODLinkSource](#) and the chapter on the data transfer in the *OpenDoc Programming Guide*.

#### New methods

The methods defined by the ODLinkSpec class include:

- [ReadLinkSpec](#)
- [WriteLinkSpec](#)

#### Overridden methods

There are currently no methods overridden by the ODLinkSpec class.

## ReadLinkSpec

## ReadLinkSpec - Syntax

This method initializes this empty link specification by reading its data from the specified focused storage unit.

```
#define INCL_ODLINKSPEC
#define INCL_ODAPI
#include <os2.h>

ODStorageUnit *su;

ReadLinkSpec(su);
```

## ReadLinkSpec Parameter - su

**su** (ODStorageUnit \*) - input

A reference to the storage unit whose focused value contains the link-specification value.

## ReadLinkSpec - Return Value

None.

## ReadLinkSpec - Parameters

**su** (ODStorageUnit \*) - input

A reference to the storage unit whose focused value contains the link-specification value.

None.

-----

## ReadLinkSpec - Remarks

If your part is a destination part, you create an empty link specification by calling the draft's [CreateLinkSpec](#) method, passing null for the source part and data parameters. Then, call the empty link specification's ReadLinkSpec method to initialize the link specification from data in the content storage unit of the clipboard or the drag-and-drop object. The storage unit should be focused on the value type kODLinkSpec in the kODPropLinkSpec property.

-----

## ReadLinkSpec - Exception Handling

kODErrCorruptLinkSpecValue

The focused storage unit contains an invalid link-specification.

kODErrNoLinkSpecValue

The focused property does not contain a link-specification value.

kODErrOutOfMemory

There is not enough memory to read the link specification.

kODErrUnknownLinkSpecVersion

The link-specification version is not recognized.

-----

## ReadLinkSpec - Related Methods

### Related Methods

- [ODDraft::CreateLinkSpec](#)
- [ODLinkSpec::WriteLinkSpec](#)

-----

## ReadLinkSpec - Topics

### Class:

ODLinkSpec

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

-----

# WriteLinkSpec

---

## WriteLinkSpec - Syntax

This method writes the data for this link specification into the specified prefocussed storage unit.

```
#define INCL_ODLINKSPEC
#define INCL_ODAPI
#include <os2.h>

ODStorageUnit      *su;

WriteLinkSpec(su);
```

---

## WriteLinkSpec Parameter - su

**su** (ODStorageUnit \*) - input  
A reference to the storage unit where the link specification data is to be written.

---

## WriteLinkSpec - Return Value

None.

---

## WriteLinkSpec - Parameters

**su** (ODStorageUnit \*) - input  
A reference to the storage unit where the link specification data is to be written.

None.

---

## WriteLinkSpec - Remarks

If your part is a source part that supports linking, you call this method to write a link specification to the content storage unit of the clipboard or

the drag-and-drop object. The storage unit should be focused on the `kODPropLinkSpec` property. This method writes the link specification to the value of type `kODLinkSpec` in the focused property, replacing any link specification that was previously stored in that value or creating the value if it does not already exist.

---

## WriteLinkSpec - Exception Handling

`kODErrOutOfMemory`

There is not enough memory to write the link specification.

---

## WriteLinkSpec - Related Methods

### Related Methods

- [ODLinkSpec::ReadLinkSpec](#)
- 

## WriteLinkSpec - Topics

### Class:

`ODLinkSpec`

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

## ODMenuBar

**Class Definition File:** `MENUBAR.IDL`

### Class Hierarchy

`SOMObject`

`ODObject`

`ODRefCntObject`

`ODBaseMenuBar`

**`ODMenuBar`**

### Description

An object of the `ODMenuBar` class represents a composite menu-bar object, made up of menus from the document shell and the active part.

When OpenDoc first opens a document, the document shell creates a menu-bar object, adds menus to it, and installs it as the *base menu-bar* object, which contains the default set of menus shared by all parts in the document. Part editors can obtain a copy of the base menu-bar object, by calling the window-state object's [CopyBaseMenuBar](#) method, add menus to it, and install it as the *current menu-bar* object.

Your part creates an empty menu-bar by calling the window-state object's [CopyBaseMenuBar](#) method. Your part can also create a copy of an existing menu-bar object by calling its menu-bar object's [Copy](#) method. These methods return a reference to the menu bar object.

OpenDoc allows the document shell and part editors to register command IDs for menu items. If no command ID is registered, a **synthetic**

command ID, one that is manufactured from the menu and menu item IDs, is generated. If a command ID is not registered and not synthetic, a part should not handle it.

## Methods

The methods defined by the ODMenuBar class include:

- [AddMenuBefore](#)
- [AddMenuItemBefore](#)
- [AddMenuItemLast](#)
- [AddMenuLast](#)
- [AddToAccelTable](#)
- [CheckMenuItem](#)
- [Copy](#)
- [Display](#)
- [EnableMenuItem](#)
- [Exists](#)
- [GetMenu](#)
- [GetMenuItem](#)
- [GetMenuItemStatusText](#)
- [GetMenuItemText](#)
- [InsertSubmenu](#)
- [IsItemChecked](#)
- [IsItemEnabled](#)
- [RemoveMenu](#)
- [RemoveMenuItem](#)
- [RestoreAccelTable](#)
- [SetHideMenuItem](#)
- [SetMenuItemStatusText](#)
- [SetMenuItemText](#)
- [SetShowMenuItem](#)

## Overridden Methods

There are currently no methods overridden by the ODMenuBar class.

# AddMenuBefore

## AddMenuBefore - Syntax

This method inserts a new menu before another specified menu on this menu bar.

```
#define INCL_ODMENUBAR
#define INCL_ODAPI
#include <os2.h>

ODMenuID      menuID;
ODPlatformMenu menu;
ODPart        *part;
ODMenuID      beforeID;

AddMenuBefore(menuID, menu, part, beforeID);
```

## AddMenuBefore Parameter - menuID

**menuID** (ODMenuID) - input  
The platform-specific ID of the new menu.

---

## AddMenuBefore Parameter - menu

**menu** (ODPlatformMenu) - input  
A platform-specific menu handle for the specified menu ID.

---

## AddMenuBefore Parameter - part

**part** (ODPart \*) - input  
A reference to the part that owns the menu or kODNULL if the menu is a document shell menu.

---

## AddMenuBefore Parameter - beforeID

**beforeID** (ODMenuID) - input  
The menu ID of the menu that follows the new menu.

---

## AddMenuBefore - Return Value

None.

---

## AddMenuBefore - Parameters

**menuID** (ODMenuID) - input  
The platform-specific ID of the new menu.

**menu** (ODPlatformMenu) - input  
A platform-specific menu handle for the specified menu ID.

**part** (ODPart \*) - input  
A reference to the part that owns the menu or kODNULL if the menu is a document shell menu.

**beforeID** (ODMenuID) - input  
The menu ID of the menu that follows the new menu.

None.

---

## AddMenuBefore - Exception Handling

kODErrOutOfMemory

There is not enough memory to allocate the menu.

---

## AddMenuBefore - Topics

### Class:

ODMenuBar

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Exception Handling](#)

---

## AddMenuItemBefore (OS/2)

---

## AddMenuItemBefore (OS/2) - Syntax

This method inserts a menu item in the specified menu before the specified menu item.

```
#define INCL_ODMENUBAR
#define INCL_ODAPI
#include <os2.h>

ODMenuID          menuID;
ODMenuItemID      subMenuID;
ODPlatformMenuItem *menuItem;
ODMenuItemID      beforeID;
ODBoolean          rv;

rv = AddMenuItemBefore(menuID, subMenuID,
    menuItem, beforeID);
```

---

## AddMenuItemBefore (OS/2) Parameter - menuID

**menuID** ([ODMenuID](#)) - input

The ID of the menu on which the menu item is to be placed.

---

## AddMenuItemBefore (OS/2) Parameter - subMenuID

**subMenuID** ([ODMenuItemID](#)) - input

The submenu ID in cases where the new menu item is to be inserted into a cascade menu.

---

## AddMenuItemBefore (OS/2) Parameter - menuItem

**menuItem** ([ODPlatformMenuItem \\*](#)) - input

The menu structure containing the menu item information.

---

## AddMenuItemBefore (OS/2) Parameter - beforeID

**beforeID** ([ODMenuItemID](#)) - input

The menu item before which the new item is to be inserted.

---

## AddMenuItemBefore (OS/2) Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating the success.

kODTrue	Successful completion
kODFalse	Error occurred

---

## AddMenuItemBefore (OS/2) - Parameters

**menuID** ([ODMenuID](#)) - input

The ID of the menu on which the menu item is to be placed.

**subMenuID** ([ODMenuItemID](#)) - input

The submenu ID in cases where the new menu item is to be inserted into a cascade menu.

**menuItem** ([ODPlatformMenuItem \\*](#)) - input

The menu structure containing the menu item information.

**beforeID** ([ODMenuItemID](#)) - input

The menu item before which the new item is to be inserted.

**rv** ([ODBoolean](#)) - returns

A flag indicating the success.

kODTrue



kODFalse	Successful completion
	Error occurred

---

## AddMenuItemBefore (OS/2) - Remarks

The value of *subMenuID* determines where the new menu is to be inserted in the specified menu, either as an item in the first level pull-down or as an item in the second level cascade. If *subMenuID* is set to kODNULL, the new menu item is inserted in *menuID* before *beforeID*. If *subMenuID* contains a value, the menu is inserted into in menu item's cascade menu before *beforeID*. The value of *beforeID* determines its position in that submenu.

---

## AddMenuItemBefore (OS/2) - Topics

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

## AddMenuItemLast (OS/2)

---

## AddMenuItemLast (OS/2) - Syntax

This method inserts a menu item at the end of the specified menu.

```
#define INCL_ODMENUBAR
#define INCL_ODAPI
#include <os2.h>

ODMenuID          menuID;
ODMenuItemID      subMenuID;
ODPlatformMenuItem *menuItem;
ODBoolean          rv;

rv = AddMenuItemLast(menuID, subMenuID, menuItem);
```

---

## AddMenuItemLast (OS/2) Parameter - menuID

**menuID** ([ODMenuID](#)) - input

The ID of the menu on which the menu item is to be placed.

---

# AddMenuItemLast (OS/2) Parameter - subMenuID

**subMenuID** (ODMenuItemID) - input

The submenu ID in cases where the new menu item is to be inserted into a cascade menu.

---

# AddMenuItemLast (OS/2) Parameter - menuItem

**menuItem** (ODPlatformMenuItem \*) - input

The menu structure containing the menu-item information.

---

# AddMenuItemLast (OS/2) Return Value - rv

**rv** (ODBoolean) - returns

A flag indicating the success.

kODTrue

Successful completion

kODFalse

Error occurred

---

# AddMenuItemLast (OS/2) - Parameters

**menuID** (ODMenuID) - input

The ID of the menu on which the menu item is to be placed.

**subMenuID** (ODMenuItemID) - input

The submenu ID in cases where the new menu item is to be inserted into a cascade menu.

**menuItem** (ODPlatformMenuItem \*) - input

The menu structure containing the menu-item information.

**rv** (ODBoolean) - returns

A flag indicating the success.

kODTrue

Successful completion

kODFalse

Error occurred

---

# AddMenuItemLast (OS/2) - Remarks

The value of *subMenuID* determines where the new menu is to be inserted in the specified menu, either as an item in the first level pull-down or as an item in the second level cascade. If *subMenuID* is set to kODNULL, the new menu item is inserted at the end of *menuID* . If

*subMenuID* contains a value, the menu is inserted at the end of the menu item's cascade menu.

---

## AddMenuItemLast (OS/2) - Topics

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

## AddMenuLast

---

## AddMenuLast - Syntax

This method appends a new menu to the end of this menu bar.

```
#define INCL_ODMENUBAR
#define INCL_ODAPI
#include <os2.h>

ODMenuID      menuID;
ODPlatformMenu menu;
ODPart        *part;

AddMenuLast(menuID, menu, part);
```

---

## AddMenuLast Parameter - menuID

**menuID** ([ODMenuID](#)) - input  
The platform-specific ID of the new menu.

---

## AddMenuLast Parameter - menu

**menu** ([ODPlatformMenu](#)) - input  
A platform-specific menu handle for the specified menu ID.

---

## AddMenuLast Parameter - part

**part** (ODPart \*) - input

A reference to a part that owns the menu or kODNULL if the menu is a document shell menu.

-----

## AddMenuLast - Return Value

None.

-----

## AddMenuLast - Parameters

**menuID** (ODMenuID) - input

The platform-specific ID of the new menu.

**menu** (ODPlatformMenu) - input

A platform-specific menu handle for the specified menu ID.

**part** (ODPart \*) - input

A reference to a part that owns the menu or kODNULL if the menu is a document shell menu.

None.

-----

## AddMenuLast - Exception Handling

kODErrOutOfMemory

There is not enough memory to allocate the menu.

-----

## AddMenuLast - Topics

**Class:**

ODMenuBar

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Exception Handling](#)

-----

## AddToAccelTable (OS/2)

---

## AddToAccelTable (OS/2) - Syntax

This method adds accelerators to the accelerator table.

```
#define INCL_ODMENUBAR
#define INCL_ODAPI
#include <os2.h>

ODULong      ulNumberOfAccels;
ODACCEL      *NewAccels;
ODBoolean     rv;

rv = AddToAccelTable(ulNumberOfAccels, NewAccels);
```

---

## AddToAccelTable (OS/2) Parameter - ulNumberOfAccels

**ulNumberOfAccels** (ODULong) - input

The number of accelerators to be added to the default table. A maximum of 20 accelerators can be added at one time.

---

## AddToAccelTable (OS/2) Parameter - NewAccels

**NewAccels** (ODACCEL \*) - input

A structure containing the accelerators and their attributes.

---

## AddToAccelTable (OS/2) Return Value - rv

**rv** (ODBoolean) - returns

A flag indicating the success.

kODTrue	Successful completion
kODFalse	Error occurred

---

## AddToAccelTable (OS/2) - Parameters

**ulNumberOfAccels** (ODULong) - input

The number of accelerators to be added to the default table. A maximum of 20 accelerators can be added at one time.

**NewAccels** (ODACCEL \*) - input

A structure containing the accelerators and their attributes.

**rv** ([ODBoolean](#)) - returns  
A flag indicating the success.

kODTrue	Successful completion
kODFalse	Error occurred

---

## AddToAccelTable (OS/2) - Topics

**Class:**  
[ODMenuBar](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## CheckMenuItem (OS/2)

---

## CheckMenuItem (OS/2) - Syntax

This method checks or unchecks the specified menu item in the menu.

```
#define INCL_ODMENUBAR
#define INCL_ODAPI
#include <os2.h>

ODMenuID      menuID;
ODMenuItemID  menuItemID;
ODBoolean     check;
ODBoolean     rv;

rv = CheckMenuItem(menuID, menuItemID, check);
```

---

## CheckMenuItem (OS/2) Parameter - menuID

**menuID** ([ODMenuID](#)) - input  
The ID of the menu containing the specified menu item.

---

## CheckMenuItem (OS/2) Parameter - menuItemID

**menuItemID** ([ODMenuItemID](#)) - input  
The ID of the menu item to be checked or unchecked.

---

## CheckMenuItem (OS/2) Parameter - check

**check** ([ODBoolean](#)) - input  
A flag indicating whether the menu item is to be checked.

kODTrue	The menu item is to be checked.
kODFalse	The menu item is to be unchecked.

---

## CheckMenuItem (OS/2) Return Value - rv

**rv** ([ODBoolean](#)) - returns  
A flag indicating the success.

kODTrue	Successful completion
kODFalse	Error occurred

---

## CheckMenuItem (OS/2) - Parameters

**menuID** ([ODMenuID](#)) - input  
The ID of the menu containing the specified menu item.

**menuItemID** ([ODMenuItemID](#)) - input  
The ID of the menu item to be checked or unchecked.

**check** ([ODBoolean](#)) - input  
A flag indicating whether the menu item is to be checked.

kODTrue	The menu item is to be checked.
kODFalse	The menu item is to be unchecked.

**rv** ([ODBoolean](#)) - returns  
A flag indicating the success.

kODTrue	Successful completion
kODFalse	Error occurred

---

## CheckMenuItem (OS/2) - Topics

**Class:**

ODMenuBar

Select an item:

[Syntax](#)[Parameters](#)[Returns](#)

---

## Copy

---

## Copy - Syntax

This method copies this menu-bar object.

```
#define INCL_ODMENUBAR
#define INCL_ODAPI
#include <os2.h>
```

```
ODMenuBar      *rv;
```

```
rv = Copy();
```

---

## Copy Return Value - rv

**rv** (ODMenuBar \*) - returns

A reference to a copy of this menu-bar object.

---

## Copy - Parameters

**rv** (ODMenuBar \*) - returns

A reference to a copy of this menu-bar object.

---

## Copy - Remarks

OpenDoc calls this method; this method is not called by most parts.

---

## Copy - Topics



**Class:**

ODMenuBar

Select an item:

[Syntax](#)[Parameters](#)[Returns](#)[Remarks](#)

---

## Display

---

## Display - Syntax

This method installs this menu-bar object as the current menu-bar, making it visible and active.

```
#define INCL_ODMENUBAR
#define INCL_ODAPI
#include <os2.h>
```

```
Display();
```

---

## Display - Return Value

None.

---

## Display - Parameters

None.

---

## Display - Topics

**Class:**

ODMenuBar

Select an item:

[Syntax](#)

---

## EnableMenuItem (OS/2)

---

## EnableMenuItem (OS/2) - Syntax

This method enables or disables the specified menu item.

```
#define INCL_ODMENUBAR
#define INCL_ODAPI
#include <os2.h>

ODMenuID      menuID;
ODMenuItemID  menuItemID;
ODBoolean     enable;
ODBoolean     rv;

rv = EnableMenuItem(menuID, menuItemID, enable);
```

---

## EnableMenuItem (OS/2) Parameter - menuID

**menuID** ([ODMenuID](#)) - input  
The ID of the menu containing the specified menu item.

---

## EnableMenuItem (OS/2) Parameter - menuItemID

**menuItemID** ([ODMenuItemID](#)) - input  
The ID of the menu item to be enabled or disabled.

---

## EnableMenuItem (OS/2) Parameter - enable

**enable** ([ODBoolean](#)) - input  
A flag indicating whether the menu item is to be enabled.

kODTrue	The menu item is to be enabled.
kODFalse	The menu item is to be disabled.

---

## EnableMenuItem (OS/2) Return Value - rv

**rv** ([ODBoolean](#)) - returns  
A flag indicating the success.

kODTrue	Successful completion
kODFalse	Error occurred

---

## EnableMenuItem (OS/2) - Parameters

**menuID** ([ODMenuID](#)) - input  
The ID of the menu containing the specified menu item.

**menuItemID** ([ODMenuItemID](#)) - input  
The ID of the menu item to be enabled or disabled.

**enable** ([ODBoolean](#)) - input  
A flag indicating whether the menu item is to be enabled.

kODTrue	The menu item is to be enabled.
kODFalse	The menu item is to be disabled.

**rv** ([ODBoolean](#)) - returns  
A flag indicating the success.

kODTrue	Successful completion
kODFalse	Error occurred

---

## EnableMenuItem (OS/2) - Topics

**Class:**  
ODMenuBar

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## Exists (OS/2)

---

# Exists (OS/2) - Syntax

This method indicates whether the specified menu item exists.

```
#define INCL_ODMENUBAR
#define INCL_ODAPI
#include <os2.h>

ODMenuID      menuID;
ODMenuItemID  menuItemID;
ODBoolean     rv;

rv = Exists(menuID, menuItemID);
```

---

## Exists (OS/2) Parameter - menuID

**menuID** (ODMenuID) - input

The ID of the menu containing the specified menu item.

---

## Exists (OS/2) Parameter - menuItemID

**menuItemID** (ODMenuItemID) - input

The ID of the menu item to be found.

---

## Exists (OS/2) Return Value - rv

**rv** (ODBoolean) - returns

A flag indicating whether the menu item exists.

kODTrue

The menu item exists.

kODFalse

The menu item does not exist.

---

## Exists (OS/2) - Parameters

**menuID** (ODMenuID) - input

The ID of the menu containing the specified menu item.

**menuItemID** (ODMenuItemID) - input

The ID of the menu item to be found.

**rv** (ODBoolean) - returns

A flag indicating whether the menu item exists.

kODTrue

The menu item exists.

kODFalse

The menu item does not exist.

-----

## Exists (OS/2) - Topics

### Class:

ODMenuBar

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

-----

## GetMenu

-----

## GetMenu - Syntax

This method returns a platform-specific menu structure for the specified menu ID.

```
#define INCL_ODMENUBAR
#define INCL_ODAPI
#include <os2.h>
```

```
ODMenuID      menu;
ODPlatformMenu rv;
```

```
rv = GetMenu(menu);
```

-----

## GetMenu Parameter - menu

**menu** ([ODMenuID](#)) - input

The platform-specific ID of a menu being requested or kODNULL if a handle to the base menu bar is to be returned.

-----

## GetMenu Return Value - rv

**rv** ([ODPlatformMenu](#)) - returns

A platform-specific menu handle for the specified menu ID.

---

## GetMenu - Parameters

**menu** ([ODMenuID](#)) - input

The platform-specific ID of a menu being requested or KODNULL if a handle to the base menu bar is to be returned.

**rv** ([ODPlatformMenu](#)) - returns

A platform-specific menu handle for the specified menu ID.

---

## GetMenu - Topics

### Class:

[ODMenuBar](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

---

## GetMenuItem (OS/2)

---

## GetMenuItem (OS/2) - Syntax

This method returns a platform-specific menu structure for the specified menu item ID.

```
#define INCL_ODMENUBAR
#define INCL_ODAPI
#include <os2.h>
```

```
ODMenuID          menuItem;
ODMenuItemID      menuItemID;
ODPlatformMenuItem *menuItem;
ODBoolean         rv;
```

```
rv = GetMenuItem(menuID, menuItemID, menuItem);
```

---

## GetMenuItem (OS/2) Parameter - menuID

**menuID** ([ODMenuID](#)) - input

The ID of the menu containing the specified menu item.

---

## GetMenuItem (OS/2) Parameter - menuItemID

**menuItemID** ([ODMenuItemID](#)) - input  
The ID of the menu item whose information is to be returned.

---

## GetMenuItem (OS/2) Parameter - menuItem

**menuItem** ([ODPlatformMenuItem \\*](#)) - output  
A menu-item structure containing the information of the specified menu-item ID.

---

## GetMenuItem (OS/2) Return Value - rv

**rv** ([ODBoolean](#)) - returns  
A flag indicating whether the item was successfully retrieved.

kODTrue	The item was successfully retrieved.
kODFalse	The item was not retrieved.

---

## GetMenuItem (OS/2) - Parameters

**menuID** ([ODMenuID](#)) - input  
The ID of the menu containing the specified menu item.

**menuItemID** ([ODMenuItemID](#)) - input  
The ID of the menu item whose information is to be returned.

**menuItem** ([ODPlatformMenuItem \\*](#)) - output  
A menu-item structure containing the information of the specified menu-item ID.

**rv** ([ODBoolean](#)) - returns  
A flag indicating whether the item was successfully retrieved.

kODTrue	The item was successfully retrieved.
kODFalse	The item was not retrieved.

---

## GetMenuItem (OS/2) - Topics

**Class:**

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

---

## GetMenuItemStatusText (OS/2)

---

### GetMenuItemStatusText (OS/2) - Syntax

This method returns the status-line text associated with the specified menu ID.

```
#define INCL_ODMENUBAR
#define INCL_ODAPI
#include <os2.h>

ODMenuID    menuID;
string      *menuStatusText;

GetMenuItemStatusText(menuID, menuStatusText);
```

---

### GetMenuItemStatusText (OS/2) Parameter - menuID

**menuID** ([ODMenuID](#)) - input

The ID of the menu whose text is to be returned.

---

### GetMenuItemStatusText (OS/2) Parameter - menuStatusText

**menuStatusText** ([string](#) \*) - output

A string containing status text of the specified menu.

---

### GetMenuItemStatusText (OS/2) - Return Value

None.

---



# GetMenuItemStatusText (OS/2) - Parameters

**menuID** ([ODMenuID](#)) - input

The ID of the menu whose text is to be returned.

**menuStatusText** ([string](#) \*) - output

A string containing status text of the specified menu.

None.

---

# GetMenuItemStatusText (OS/2) - Topics

**Class:**

[ODMenuBar](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

---

# GetMenuItemText (OS/2)

---

# GetMenuItemText (OS/2) - Syntax

This method returns text associated with the specified menu item.

```
#define INCL_ODMENUBAR
#define INCL_ODAPI
#include <os2.h>

ODMenuID      menuID;
ODMenuItemID  menuItemID;
string        *menuItemString;
ODUSHort      rv;

rv = GetMenuItemText(menuID, menuItemID, menuItemString);
```

---

# GetMenuItemText (OS/2) Parameter - menuID

**menuID** ([ODMenuID](#)) - input

The ID of the menu containing the specified menu item.

---

## GetMenuItemText (OS/2) Parameter - menuItemID

**menuItemID** ([ODMenuItemID](#)) - input  
The ID of the menu item whose text is to be returned.

---

## GetMenuItemText (OS/2) Parameter - menuItemString

**menuItemString** ([string \\*](#)) - output  
A string containing text of the specified menu item.

---

## GetMenuItemText (OS/2) Return Value - rv

**rv** ([ODUShort](#)) - returns  
The length of the returned string.

---

## GetMenuItemText (OS/2) - Parameters

**menuID** ([ODMenuID](#)) - input  
The ID of the menu containing the specified menu item.

**menuItemID** ([ODMenuItemID](#)) - input  
The ID of the menu item whose text is to be returned.

**menuItemString** ([string \\*](#)) - output  
A string containing text of the specified menu item.

**rv** ([ODUShort](#)) - returns  
The length of the returned string.

---

## GetMenuItemText (OS/2) - Topics

**Class:**  
ODMenuBar

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

# InsertSubmenu (OS/2)

---

## InsertSubmenu (OS/2) - Syntax

This method inserts a new submenu in the specified menu or submenu, creating a cascading menu.

```
#define INCL_ODMENUBAR
#define INCL_ODAPI
#include <os2.h>

ODMenuID      menuID;
ODMenuItemID  subMenuID;
ODPlatformMenu newSubMenu;
ODBoolean     rv;

rv = InsertSubmenu(menuID, subMenuID, newSubMenu);
```

---

## InsertSubmenu (OS/2) Parameter - menuID

**menuID** ([ODMenuID](#)) - input  
The ID of the menu on which the new submenu item is to be inserted.

---

## InsertSubmenu (OS/2) Parameter - subMenuID

**subMenuID** ([ODMenuItemID](#)) - input  
The ID of the menu item to which this submenu is to be attached. If this value is KODNULL, the submenu is to be inserted into *menuID*. This replaces any existing submenu in the *menuID* parameter.

---

## InsertSubmenu (OS/2) Parameter - newSubMenu

**newSubMenu** ([ODPlatformMenu](#)) - input  
The menu handle containing the menu-item information.

---

## InsertSubmenu (OS/2) Return Value - rv

**rv** ([ODBoolean](#)) - returns  
A flag indicating the success.

kODTrue	Successful completion
kODFalse	Error occurred

-----

## InsertSubmenu (OS/2) - Parameters

**menuID** ([ODMenuID](#)) - input  
The ID of the menu on which the new submenu item is to be inserted.

**subMenuID** ([ODMenuItemID](#)) - input  
The ID of the menu item to which this submenu is to be attached. If this value is kODNULL, the submenu is to be inserted into *menuID* . This replaces any existing submenu in the *menuID* parameter.

**newSubMenu** ([ODPlatformMenu](#)) - input  
The menu handle containing the menu-item information.

**rv** ([ODBoolean](#)) - returns  
A flag indicating the success.

kODTrue	Successful completion
kODFalse	Error occurred

-----

## InsertSubmenu (OS/2) - Remarks

The value of *subMenuID* determines where the new submenu is to be inserted in the menu. If *subMenuID* contains a kODNULL, the new submenu is inserted in *menuID* . If *subMenuID* contains a value, the submenu is inserted into *subMenuID* .

-----

## InsertSubmenu (OS/2) - Topics

**Class:**  
ODMenuBar

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

-----

## IsValid (OS/2)

-----

## IsValid (OS/2) - Syntax

This method indicates whether this menu bar object is equivalent to the base menu bar object from which it was copied.

```
#define INCL_ODMENUBAR
#define INCL_ODAPI
#include <os2.h>

ODBoolean    rv;

rv = IsValid();
```

---

## IsValid (OS/2) Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating whether this menu bar object is equivalent to the base menu bar object from which it was copied.

kODTrue	The menu bar objects are not equivalent.
kODFalse	The menu bar objects are equivalent.

---

## IsValid (OS/2) - Parameters

**rv** ([ODBoolean](#)) - returns

A flag indicating whether this menu bar object is equivalent to the base menu bar object from which it was copied.

kODTrue	The menu bar objects are not equivalent.
kODFalse	The menu bar objects are equivalent.

---

## IsValid (OS/2) - Remarks

Your part calls its cached base menu bar's `IsValid` method before adding its own menus and displaying the composite menu bar. If this base menu bar is no longer valid, your part recopies the current base menu bar object by calling its window state's [CopyBaseMenuBar](#) method.

---

## IsValid (OS/2) - Related Methods

### Related Methods

- [ODWindowState::CopyBaseMenuBar](#)

---

## IsValid (OS/2) - Topics

**Class:**

ODMenuBar

Select an item:

[Syntax](#)[Parameters](#)[Returns](#)[Remarks](#)[Related Methods](#)

---

## IsItemChecked (OS/2)

---

### IsItemChecked (OS/2) - Syntax

This method indicates whether a menu item is checked.

```
#define INCL_ODMENUBAR
#define INCL_ODAPI
#include <os2.h>

ODMenuID      menuID;
ODMenuItemID  menuItemID;
ODBoolean     rv;

rv = IsItemChecked(menuID, menuItemID);
```

---

### IsItemChecked (OS/2) Parameter - menuID

**menuID** ([ODMenuID](#)) - input

The ID of the menu containing the specified menu item.

---

### IsItemChecked (OS/2) Parameter - menuItemID

**menuItemID** ([ODMenuItemID](#)) - input

The ID of the menu item to be queried.

---

### IsItemChecked (OS/2) Return Value - rv

**rv** ([ODBoolean](#)) - returns  
A flag indicating whether the specified menu item is checked.

<b>kODTrue</b>	The specified menu item is checked.
<b>kODFalse</b>	The specified menu item is not checked.

---

## IsItemChecked (OS/2) - Parameters

**menuID** ([ODMenuID](#)) - input  
The ID of the menu containing the specified menu item.

**menuItemID** ([ODMenuItemID](#)) - input  
The ID of the menu item to be queried.

**rv** ([ODBoolean](#)) - returns  
A flag indicating whether the specified menu item is checked.

<b>kODTrue</b>	The specified menu item is checked.
<b>kODFalse</b>	The specified menu item is not checked.

---

## IsItemChecked (OS/2) - Topics

**Class:**  
ODMenuBar

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## IsItemEnabled (OS/2)

---

## IsItemEnabled (OS/2) - Syntax

This method indicates whether a menu item is enabled.

```
#define INCL_ODMENUBAR
#define INCL_ODAPI
#include <os2.h>

ODMenuID      menuItem;
ODMenuItemID  menuItemID;
ODBoolean     rv;

rv = IsItemEnabled(menuID, menuItemID);
```

---

## IsItemEnabled (OS/2) Parameter - menuID

**menuID** ([ODMenuID](#)) - input

The ID of the menu containing the specified menu item.

---

## IsItemEnabled (OS/2) Parameter - menuItemID

**menuItemID** ([ODMenuItemID](#)) - input

The ID of the menu item to be queried.

---

## IsItemEnabled (OS/2) Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the specified menu item is enabled.

kODTrue

The specified menu item is enabled.

kODFalse

The specified menu item is disabled.

---

## IsItemEnabled (OS/2) - Parameters

**menuID** ([ODMenuID](#)) - input

The ID of the menu containing the specified menu item.

**menuItemID** ([ODMenuItemID](#)) - input

The ID of the menu item to be queried.

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the specified menu item is enabled.

kODTrue

The specified menu item is enabled.

kODFalse

The specified menu item is disabled.

---

## IsItemEnabled (OS/2) - Topics

**Class:**



ODMenuBar

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

---

## RemoveMenu

---

## RemoveMenu - Syntax

This method removes the menu with the specified ID from this menu bar.

```
#define INCL_ODMENUBAR
#define INCL_ODAPI
#include <os2.h>
```

```
ODMenuID    menu;
```

```
RemoveMenu (menu) ;
```

---

## RemoveMenu Parameter - menu

**menu** ([ODMenuID](#)) - input

The platform-specific ID of the menu to be removed.

---

## RemoveMenu - Return Value

None.

---

## RemoveMenu - Parameters

**menu** ([ODMenuID](#)) - input

The platform-specific ID of the menu to be removed.

None.

---

## RemoveMenu - Remarks

The displayed menu bar is not affected.

---

## RemoveMenu - Topics

### Class:

ODMenuBar

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

---

## RemoveMenuItem (OS/2)

---

## RemoveMenuItem (OS/2) - Syntax

This method removes the specified menu item from the menu bar.

```
#define INCL_ODMENUBAR
#define INCL_ODAPI
#include <os2.h>
```

```
ODMenuID      menuID;
ODMenuItemID  menuItemID;
ODBoolean     rv;
```

```
rv = RemoveMenuItem(menuID, menuItemID);
```

---

## RemoveMenuItem (OS/2) Parameter - menuID

**menuID** ([ODMenuID](#)) - input

The ID of the menu containing the specified menu item.

---

## RemoveMenuItem (OS/2) Parameter - menuItemID

**menuItemID** ([ODMenuItemID](#)) - input  
The ID of the menu item to be removed.

---

## RemoveMenuItem (OS/2) Return Value - rv

**rv** ([ODBoolean](#)) - returns  
A flag indicating whether the item was successfully removed.

kODTrue	The item was successfully removed.
kODFalse	The item was not removed.

---

## RemoveMenuItem (OS/2) - Parameters

**menuID** ([ODMenuID](#)) - input  
The ID of the menu containing the specified menu item.

**menuItemID** ([ODMenuItemID](#)) - input  
The ID of the menu item to be removed.

**rv** ([ODBoolean](#)) - returns  
A flag indicating whether the item was successfully removed.

kODTrue	The item was successfully removed.
kODFalse	The item was not removed.

---

## RemoveMenuItem (OS/2) - Topics

**Class:**  
ODMenuBar

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## RestoreAccelTable (OS/2)

---

## RestoreAccelTable (OS/2) - Syntax

This method restores the OpenDoc accelerator table to its initial state and removes any part-defined accelerators.

```
#define INCL_ODMENUBAR
#define INCL_ODAPI
#include <os2.h>
```

```
RestoreAccelTable();
```

---

## RestoreAccelTable (OS/2) - Return Value

None.

---

## RestoreAccelTable (OS/2) - Parameters

None.

---

## RestoreAccelTable (OS/2) - Topics

**Class:**  
ODMenuBar

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## SetHideMenuItem (OS/2)

---

## SetHideMenuItem (OS/2) - Syntax

This method hides the text of the specified menu item.

```
#define INCL_ODMENUBAR
#define INCL_ODAPI
#include <os2.h>
```

```
ODMenuID      menuID;
ODMenuItemID  menuItemID;
ODBoolean     rv;

rv = SetHideMenuItem(menuID, menuItemID);
```

---

## SetHideMenuItem (OS/2) Parameter - menuID

**menuID** ([ODMenuID](#)) - input  
The ID of the menu containing the specified menu item.

---

## SetHideMenuItem (OS/2) Parameter - menuItemID

**menuItemID** ([ODMenuItemID](#)) - input  
The ID of the menu item whose text is to be hidden. This parameter can be set to one of the following values:

VIEW\_SHOWFRAMEOUTLINE  
Hide the **Hide/Show Frame Outline** menu item.  
VIEW\_SHOWLINKS  
Hide the **Hide/Show Links** menu item.

---

## SetHideMenuItem (OS/2) Return Value - rv

**rv** ([ODBoolean](#)) - returns  
A flag indicating the success.

kODTrue           Successful completion  
kODFalse          Error occurred

---

## SetHideMenuItem (OS/2) - Parameters

**menuID** ([ODMenuID](#)) - input  
The ID of the menu containing the specified menu item.

**menuItemID** ([ODMenuItemID](#)) - input  
The ID of the menu item whose text is to be hidden. This parameter can be set to one of the following values:

VIEW\_SHOWFRAMEOUTLINE  
Hide the **Hide/Show Frame Outline** menu item.  
VIEW\_SHOWLINKS  
Hide the **Hide/Show Links** menu item.

**rv** ([ODBoolean](#)) - returns  
A flag indicating the success.

kODTrue	Successful completion
kODFalse	Error occurred

---

## SetHideMenuItem (OS/2) - Topics

**Class:**  
ODMenuBar

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## SetMenuItemStatusText (OS/2)

---

## SetMenuItemStatusText (OS/2) - Syntax

This method sets the status-line text for the specified menu ID.

```
#define INCL_ODMENUBAR
#define INCL_ODAPI
#include <os2.h>

ODMenuID    menuID;
string      menuStatusText;

SetMenuItemStatusText(menuID, menuStatusText);
```

---

## SetMenuItemStatusText (OS/2) Parameter - menuID

**menuID** ([ODMenuID](#)) - input  
The ID of the menu whose text is to be set.

---

## SetMenuItemStatusText (OS/2) Parameter - menuStatusText

**menuStatusText** ([string](#)) - input  
A string containing status text of the specified menu.

---

## SetMenuItemStatusText (OS/2) - Return Value

None.

---

## SetMenuItemStatusText (OS/2) - Parameters

**menuID** ([ODMenuID](#)) - input

The ID of the menu whose text is to be set.

**menuStatusText** ([string](#)) - input

A string containing status text of the specified menu.

None.

---

## SetMenuItemStatusText (OS/2) - Topics

**Class:**

ODMenuBar

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

---

## SetMenuItemText (OS/2)

---

## SetMenuItemText (OS/2) - Syntax

This method sets the text associated with the specified menu item.

```
#define INCL_ODMENUBAR
#define INCL_ODAPI
#include <os2.h>
```

```
ODMenuID      menuID;
ODMenuItemID  menuItemID;
string         menuItemString;
ODBoolean     rv;
```

```
rv = SetMenuItemText(menuID, menuItemID, menuItemString);
```

---

## SetMenuItemText (OS/2) Parameter - menuID

**menuID** ([ODMenuID](#)) - input  
The ID of the menu containing the specified menu item.

---

## SetMenuItemText (OS/2) Parameter - menuItemID

**menuItemID** ([ODMenuItemID](#)) - input  
The ID of the menu item whose text is to be changed.

---

## SetMenuItemText (OS/2) Parameter - menuItemString

**menuItemString** ([string](#)) - input  
A string containing the new text to place on the menu item.

---

## SetMenuItemText (OS/2) Return Value - rv

**rv** ([ODBoolean](#)) - returns  
A flag indicating the success.

kODTrue	Successful completion
kODFalse	Error occurred

---

## SetMenuItemText (OS/2) - Parameters

**menuID** ([ODMenuID](#)) - input  
The ID of the menu containing the specified menu item.

**menuItemID** ([ODMenuItemID](#)) - input  
The ID of the menu item whose text is to be changed.

**menuItemString** ([string](#)) - input  
A string containing the new text to place on the menu item.

**rv** ([ODBoolean](#)) - returns  
A flag indicating the success.



kODTrue	Successful completion
kODFalse	Error occurred

---

## SetMenuItemText (OS/2) - Topics

**Class:**  
ODMenuBar

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## SetShowMenuItem (OS/2)

---

## SetShowMenuItem (OS/2) - Syntax

This method displays the text of the specified menu item.

```
#define INCL_ODMENUBAR
#define INCL_ODAPI
#include <os2.h>

ODMenuID      menuID;
ODMenuItemID  menuItemID;
ODBoolean     rv;

rv = SetShowMenuItem(menuID, menuItemID);
```

---

## SetShowMenuItem (OS/2) Parameter - menuID

**menuID** ([ODMenuID](#)) - input  
The ID of the menu containing the specified menu item.

---

## SetShowMenuItem (OS/2) Parameter - menuItemID

**menuItemID** ([ODMenuItemID](#)) - input  
The ID of the menu item whose text is to be displayed. This parameter can be set to one of the following values:

VIEW\_SHOWFRAMEOUTLINE  
Displays the **Hide/Show Frame Outline** menu item.  
VIEW\_SHOWLINKS  
Displays the **Hide/Show Links** menu item.

---

## SetShowMenuItem (OS/2) Return Value - rv

**rv** ([ODBoolean](#)) - returns  
A flag indicating the success.

kODTrue	Successful completion
kODFalse	Error occurred

---

## SetShowMenuItem (OS/2) - Parameters

**menuID** ([ODMenuID](#)) - input  
The ID of the menu containing the specified menu item.

**menuItemID** ([ODMenuItemID](#)) - input  
The ID of the menu item whose text is to be displayed. This parameter can be set to one of the following values:

VIEW\_SHOWFRAMEOUTLINE  
Displays the **Hide/Show Frame Outline** menu item.  
VIEW\_SHOWLINKS  
Displays the **Hide/Show Links** menu item.

**rv** ([ODBoolean](#)) - returns  
A flag indicating the success.

kODTrue	Successful completion
kODFalse	Error occurred

---

## SetShowMenuItem (OS/2) - Topics

**Class:**  
ODMenuBar

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## ODMessageInterface

**Class Definition File:** MSSGINTF.IDL

## Class Hierarchy

SOMObject  
 ODOObject  
 ODBaseMessageInterface  
 **ODMessageInterface**

## Description

An object of the ODMessageInterface class provides the capability of creating and sending semantic events. This class works with the [ODSemanticInterface](#) class to handle messaging between parts.

When a document is opened, the session object creates a single message interface object. All parts of that document share the message interface object; you can obtain a reference to it by calling the session object's [GetMessageInterface](#) method.

The OpenDoc message-interface object is responsible for constructing OSA events, getting and setting event attributes, parsing events, and sending events. OpenDoc supports your part editor's ability to create and send semantic events through another object, the message interface, and to other parts. Your part can also send semantic events to the document shell by specifying the constant kODAppShell for the destination part. The message interface is also the object through which OpenDoc sends semantic events to your part, although your part does not make any calls to the message interface in that situation.

For more information related to semantic events, see the class description for [ODSemanticInterface](#). For more information related to creating and sending OSA events, see the chapter on creating and sending OSA events in the *Open Scripting Architecture Guide and Reference for OS/2*. For general information on scripting support in OpenDoc, see the chapter on semantic events and scripting in the *OpenDoc Programming Guide*.

## Methods

The methods defined by the ODMessageInterface class include:

- [CreateEvent](#)
- [CreatePartAddrDesc](#)
- [CreatePartObjSpec](#)
- [ProcessSemanticEvent](#)
- [Send](#)

## Overridden Methods

There are currently no methods overridden by the ODMessageInterface class.

---

# CreateEvent

---

## CreateEvent - Syntax

This method creates an OSA event object.

```
#define INCL_ODMESSAGEINTERFACE
#define INCL_ODAPI
#include <os2.h>

OEventClass      theAEEEventClass;
OEventID         theAEEEventID;
ODAddressDesc    *target;
ODSLong          transactionID;
ODOSAEvent       **result;
ODSShort         rv;

rv = CreateEvent(theAEEEventClass, theAEEEventID,
                target, transactionID, result);
```

---

## CreateEvent Parameter - theAEEEventClass

**theAEEEventClass** ([ODEventClass](#)) - input  
The event class of the OSA event object to be created.

---

## CreateEvent Parameter - theAEEEventID

**theAEEEventID** ([ODEventID](#)) - input  
The event ID of the OSA event object to be created.

---

## CreateEvent Parameter - target

**target** ([ODAddressDesc](#) \*) - input  
A reference to an address of the destination part.

---

## CreateEvent Parameter - transactionID

**transactionID** ([ODSLong](#)) - input  
A value that uniquely identifies this transaction.

---

## CreateEvent Parameter - result

**result** ([ODOSAEvent](#) \*\*) - in/out  
The reference to the new OSA event object.

---

## CreateEvent Return Value - rv

**rv** ([ODSShort](#)) - returns  
The return ID assigned to the event.

---

## CreateEvent - Parameters



---

# CreatePartAddrDesc

---

## CreatePartAddrDesc - Syntax

This method creates an address descriptor that specifies the address of the specified part.

```
#define INCL_ODMESSAGEINTERFACE
#define INCL_ODAPI
#include <os2.h>

ODAddressDesc    **theAddressDesc;
ODPart           *part;

CreatePartAddrDesc(theAddressDesc, part);
```

---

## CreatePartAddrDesc Parameter - theAddressDesc

**theAddressDesc** (ODAddressDesc \*\*) - in/out  
A reference to the address descriptor object to be set to the address of the specified part.

---

## CreatePartAddrDesc Parameter - part

**part** (ODPart \*) - input  
A reference to the part whose address is desired.

---

## CreatePartAddrDesc - Return Value

None.

---

# CreatePartAddrDesc - Parameters

**theAddressDesc** (ODAddressDesc \*\*) - in/out

A reference to the address descriptor object to be set to the address of the specified part.

**part** (ODPart \*) - input

A reference to the part whose address is desired.

None.

---

## CreatePartAddrDesc - Remarks

This method is called to create an address descriptor that identifies the OpenDoc document in which the destination part resides.

The address descriptor returned by this method should not be stored persistently because it is only valid while the part is instantiated.

---

## CreatePartAddrDesc - Exception Handling

This method may throw platform-specific exceptions.

---

## CreatePartAddrDesc - Related Methods

### Related Methods

- [ODMessageInterface::CreatePartObjSpec](#)
- 

## CreatePartAddrDesc - Topics

### Class:

ODMessageInterface

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

## CreatePartObjSpec

---

## CreatePartObjSpec - Syntax

This method creates an object specifier that refers to the specified part.

```
#define INCL_ODMESSAGEINTERFACE
#define INCL_ODAPI
#include <os2.h>

ODObjectSpec    **theObjSpec;
ODPart          *thePart;

CreatePartObjSpec(theObjSpec, thePart);
```

---

## CreatePartObjSpec Parameter - theObjSpec

**theObjSpec** (ODObjectSpec \*\*) - in/out  
A reference to the object specifier to be set to refer to the specified part.

---

## CreatePartObjSpec Parameter - thePart

**thePart** (ODPart \*) - input  
A reference to the part for which an object specifier is to be created.

---

## CreatePartObjSpec - Return Value

None.

---

## CreatePartObjSpec - Parameters

**theObjSpec** (ODObjectSpec \*\*) - in/out  
A reference to the object specifier to be set to refer to the specified part.

**thePart** (ODPart \*) - input  
A reference to the part for which an object specifier is to be created.

None.

---



## CreatePartObjSpec - Remarks

If your part sends a semantic event to another part, you may use this method to address that part or to create an object specifier that represents that part. You must also call your message interface object's [CreatePartAddrDesc](#) method to create an address descriptor that identifies the OpenDoc document in which the destination part resides.

The object specifier created by this method has the type cPart and has a null container type. The object specifier allows your part to communicate privately with each other and should not be stored persistently because it is only valid while the part is instantiated. The object specifier should also not be used in a recordable event.

---

## CreatePartObjSpec - Exception Handling

This method may throw platform-specific exceptions.

---

## CreatePartObjSpec - Related Methods

### Related Methods

- [ODMessageInterface::CreatePartAddrDesc](#)
- 

## CreatePartObjSpec - Topics

### Class:

ODMessageInterface

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## ProcessSemanticEvent

---

## ProcessSemanticEvent - Syntax

This method dispatches an OSA event object to the appropriate handler.

```
#define INCL_ODMESSAGEINTERFACE
```

```

#define INCL_ODAPI
#include <os2.h>

ODEventData    *theEvent;
ODBoolean      rv;

rv = ProcessSemanticEvent(theEvent);

```

-----

## ProcessSemanticEvent Parameter - theEvent

**theEvent** (ODEventData \*) - input  
A platform-specific structure representing an event.

-----

## ProcessSemanticEvent Return Value - rv

**rv** (ODBoolean) - returns  
A flag indicating whether the event was processed.

kODTrue	The event was processed.
kODFalse	The event was not processed.

-----

## ProcessSemanticEvent - Parameters

**theEvent** (ODEventData \*) - input  
A platform-specific structure representing an event.

**rv** (ODBoolean) - returns  
A flag indicating whether the event was processed.

kODTrue	The event was processed.
kODFalse	The event was not processed.

-----

## ProcessSemanticEvent - Remarks

OpenDoc calls this method to dispatch OSA event objects to the appropriate part. Your part should not call this method.

-----

## ProcessSemanticEvent - Topics

**Class:**

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

## Send

## Send - Syntax

This method sends the specified OSA event object and requests a reply if appropriate.

```
#define INCL_ODMESSAGEINTERFACE
#define INCL_ODAPI
#include <os2.h>

ODFrame      *toFrame;
ODPart       *part;
ODOSAEEvent  *theOSAEvent;
ODOSAEEvent  *reply;
ODSendMode   sendMode;
ODSendPriority sendPriority;
ODULong      timeOutInTicks;

Send(toFrame, part, theOSAEvent, reply, sendMode,
     sendPriority, timeOutInTicks);
```

## Send Parameter - toFrame

**toFrame** (ODFrame \*) - input

A reference to a frame belonging to the part sending the OSA event object or kODNULL if the part sending the OSA event object has no frame.

## Send Parameter - part

**part** (ODPart \*) - input

A reference to the part sending the OSA event object or kODAppShell if the sender is the application shell or document.

## Send Parameter - theOSAEvent

**theOSAEvent** (ODOSAEvent \*) - input  
A reference to an OSA event object to be sent.

---

## Send Parameter - reply

**reply** (ODOSAEvent \*) - in/out  
A reference to a reply OSA event object to be returned.

---

## Send Parameter - sendMode

**sendMode** (ODSendMode) - input  
The flags that specify the interactions between the sending and receiving parts. This parameter can be set to one of the following values:

---

## Send Parameter - sendPriority

**sendPriority** (ODSendPriority) - input  
The priority of the event. This parameter can be set to one of the following values:

---

## Send Parameter - timeOutInTicks

**timeOutInTicks** (ODULong) - input  
The time to wait for a reply before generating a time-out exception, expressed in ticks (60ths of a second).

---

## Send - Return Value

None.

---

## Send - Parameters

**toFrame** (ODFrame \*) - input

A reference to a frame belonging to the part sending the OSA event object or kODNULL if the part sending the OSA event object has no frame.

**part** (ODPart \*) - input

A reference to the part sending the OSA event object or kODAppShell if the sender is the application shell or document.

**theOSAEvent** (ODOSAEvent \*) - input

A reference to an OSA event object to be sent.

**reply** (ODOSAEvent \*) - in/out

A reference to a reply OSA event object to be returned.

**sendMode** (ODSendMode) - input

The flags that specify the interactions between the sending and receiving parts. This parameter can be set to one of the following values:

**sendPriority** (ODSendPriority) - input

The priority of the event. This parameter can be set to one of the following values:

**timeOutInTicks** (ODULong) - input

The time to wait for a reply before generating a time-out exception, expressed in ticks (60ths of a second).

None.

-----

## Send - Remarks

Before calling this method, you must create the OSA event object to be sent using the message interface's [CreateEvent](#) method. Your part is responsible for deleting both the OSA event object and any reply that is returned.

-----

## Send - Exception Handling

This method may throw platform-specific exceptions.

-----

## Send - Related Methods

### Related Methods

- [ODMessageInterface::CreateEvent](#)
- [ODMessageInterface::CreatePartAddrDesc](#)

-----

## Send - Topics

### Class:

ODMessageInterface

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

# ODNameResolver

**Class Definition File:** NAMRSLVR.IDL

## Class Hierarchy

```
SOMObject
  ODOObject
    ODBaseNameResolve
      ODNameResolver
```

## Description

An object of the ODNameResolver class resolves object specifiers in semantic events. The ODNameResolver class works with the [ODSemanticInterface](#) class to resolve OSA event object specifiers sent between parts.

When a document is opened, the session object creates a single name-resolver object. All parts of that document share the name resolver; you can obtain a reference to it by calling the session object's [GetNameResolver](#) method. Your part's semantic interface can use the name resolver to initiate the resolution of an object specifier and to call your part's object accessors. Your part can also use the name resolver to create a *swap token*, which allows your part to pass the resolution of an object specifier to another part.

For more information related to object accessors, see the class description for [ODSemanticInterface](#). For more information related to object specifiers and the OSA event object model, see the chapter on resolving and creating object specifier records in the *Open Scripting Architecture Guide and Reference for OS/2* and the chapter on scripting in the *OpenDoc Programming Guide*.

## Methods

The methods defined by the ODNameResolver class include:

- [CallObjectAccessor](#)
- [CreateSwapToken](#)
- [DisposeToken](#)
- [GetContextFromToken](#)
- [GetUserToken](#)
- [IsODToken](#)
- [Resolve](#)

## Overridden Methods

There are currently no methods overridden by the ODNameResolver class.

---

# CallObjectAccessor

---

## CallObjectAccessor - Syntax

This method calls the object accessor of the calling part's semantic interface.

```
#define INCL_ODNAMERESOLVER
#define INCL_ODAPI
#include <os2.h>

ODPart      *part;
ODDescType  desiredClass;
ODOSLToken  *containerToken;
ODDescType  containerClass;
```

```
ODDescType      keyForm;  
ODDesc          *keyData;  
ODOSLToken      *token;
```

```
CallObjectAccessor(part, desiredClass, containerToken,  
                  containerClass, keyForm, keyData, token);
```

---

## CallObjectAccessor Parameter - part

**part** (ODPart \*) - input

A reference to the part calling this method-that is, the context for the supplied token.

---

## CallObjectAccessor Parameter - desiredClass

**desiredClass** (ODDescType) - input

The class of the desired OSA event objects.

---

## CallObjectAccessor Parameter - containerToken

**containerToken** (ODOSLToken \*) - input

A reference to the token which identifies the container for the desired objects.

---

## CallObjectAccessor Parameter - containerClass

**containerClass** (ODDescType) - input

The class of the container for the desired OSA event objects.

---

## CallObjectAccessor Parameter - keyForm

**keyForm** (ODDescType) - input

The key form specified by the object specifier record for the objects to be located.

---

## CallObjectAccessor Parameter - keyData

**keyData** (ODDesc \*) - input

A reference to the descriptor object containing the key data specified by the object specifier record for the objects to be located.

---

## CallObjectAccessor Parameter - token

**token** (ODOSLToken \*) - output

A reference to the token to be filled in by the object accessor being called.

---

## CallObjectAccessor - Return Value

None.

---

## CallObjectAccessor - Parameters

**part** (ODPart \*) - input

A reference to the part calling this method-that is, the context for the supplied token.

**desiredClass** (ODDescType) - input

The class of the desired OSA event objects.

**containerToken** (ODOSLToken \*) - input

A reference to the token which identifies the container for the desired objects.

**containerClass** (ODDescType) - input

The class of the container for the desired OSA event objects.

**keyForm** (ODDescType) - input

The key form specified by the object specifier record for the objects to be located.

**keyData** (ODDesc \*) - input

A reference to the descriptor object containing the key data specified by the object specifier record for the objects to be located.

**token** (ODOSLToken \*) - output

A reference to the token to be filled in by the object accessor being called.

None.

---

## CallObjectAccessor - Remarks

You can use this method to call the object accessor for the particular object class in a particular container. This is especially useful for constructing a list of tokens. Your part must call this method instead of directly calling the [CallObjectAccessor](#) method of your part's semantic-interface object.

The *part* parameter must be a reference to the part that calls this method.



---

# CallObjectAccessor - Exception Handling

The OSA Event Manager may throw an exception if the object accessor could not be found for the specified part.

Parts and OpenDoc can throw other exceptions.

---

## CallObjectAccessor - Related Methods

### Related Methods

- [ODSemanticInterface::CallObjectAccessor](#)
- 

## CallObjectAccessor - Topics

### Class:

[ODNameResolver](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

## CreateSwapToken

---

## CreateSwapToken - Syntax

This method creates a token that passes the resolution of an object specifier to another part.

```
#define INCL_ODNAMERESOLVER
#define INCL_ODAPI
#include <os2.h>

ODOSLToken    *token;
ODPart        *part;
ODFrame       *frame;

CreateSwapToken(token, part, frame);
```

---

## CreateSwapToken Parameter - token

**token** (ODOSLToken \*) - in/out

A reference to the token to contain the part and frame information.

---

## CreateSwapToken Parameter - part

**part** (ODPart \*) - input

A reference to the part to be specified by the token.

---

## CreateSwapToken Parameter - frame

**frame** (ODFrame \*) - input

A reference to the frame to be specified by the token or kODNULL if the specified part has no frames.

---

## CreateSwapToken - Return Value

None.

---

## CreateSwapToken - Parameters

**token** (ODOSLToken \*) - in/out

A reference to the token to contain the part and frame information.

**part** (ODPart \*) - input

A reference to the part to be specified by the token.

**frame** (ODFrame \*) - input

A reference to the frame to be specified by the token or kODNULL if the specified part has no frames.

None.

---

## CreateSwapToken - Remarks

This method fills in the empty token you provide with information about the specified part and frame.

This method is intended to be called by your part's object accessors if it cannot handle the resolution of an object specifier but know of another part that can (for instance, an embedded part). In this case, you object accessor can return a swap token to OpenDoc, which then tries to resolve the object specifier using the new part and frame.

---

## CreateSwapToken - Exception Handling

kODErrOutOfMemory

There was not enough memory to create the token data structures.

---

## CreateSwapToken - Related Methods

### Related Methods

- [ODNameResolver::GetContextFromToken](#)
  - [ODSemanticInterface::CallObjectAccessor](#)
- 

## CreateSwapToken - Topics

### Class:

[ODNameResolver](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## DisposeToken

---

## DisposeToken - Syntax

This method deallocates the internal structures of the specified token.

```
#define INCL_ODNAMERESOLVER
#define INCL_ODAPI
#include <os2.h>
```

```
ODOSLToken    *theToken;
```

```
DisposeToken(theToken);
```

---

## DisposeToken Parameter - theToken

**theToken** (ODOSLToken \*) - input  
A reference to the token to be deallocated.

---

## DisposeToken - Return Value

None.

---

## DisposeToken - Parameters

**theToken** (ODOSLToken \*) - input  
A reference to the token to be deallocated.

None.

---

## DisposeToken - Remarks

This method calls the semantic interface of the part that created the token to be disposed of the token's internal data structures. If the semantic interface's [CallDisposeTokenProc](#) method does not handle the event, this method handles the disposal of the token.

---

## DisposeToken - Exception Handling

This method may throw platform-specific exceptions.

---

## DisposeToken - Related Methods

### Related Methods

- [ODSemanticInterface::CallDisposeTokenProc](#)
-

# DisposeToken - Topics

## Class:

ODNameResolver

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

## GetContextFromToken

---

## GetContextFromToken - Syntax

This method retrieves references to the part and the frame in whose context a specified token was created.

```
#define INCL_ODNAMERESOLVER
#define INCL_ODAPI
#include <os2.h>

ODOSLToken      *token;
ODPart          **part;
ODFrame         **frame;

GetContextFromToken(token, part, frame);
```

---

## GetContextFromToken Parameter - token

**token** (ODOSLToken \*) - input

A reference to the token to be examined.

---

## GetContextFromToken Parameter - part

**part** (ODPart \*\*) - output

A reference to the part representing the context for this token.

---

## GetContextFromToken Parameter - frame

**frame** (ODFrame \*\*) - output

A reference to the frame representing the context for this token or kODNULL if the specified part does not have any frames associated with it.

-----

## GetContextFromToken - Return Value

None.

-----

## GetContextFromToken - Parameters

**token** (ODOSToken \*) - input

A reference to the token to be examined.

**part** (ODPart \*\*) - output

A reference to the part representing the context for this token.

**frame** (ODFrame \*\*) - output

A reference to the frame representing the context for this token or kODNULL if the specified part does not have any frames associated with it.

None.

-----

## GetContextFromToken - Remarks

The *token* parameter must be a valid OpenDoc token; you can call the [IsODToken](#) method to check whether a token is valid.

This method does not increment the reference count of either the part or the frame in whose context the token was created.

-----

## GetContextFromToken - Related Methods

### Related Methods

- [ODNameResolver::CreateSwapToken](#)
- [ODNameResolver::IsODToken](#)

-----

## GetContextFromToken - Topics

**Class:**

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## GetUserToken

---

### GetUserToken - Syntax

This method returns a reference to the descriptor object contained in the specified OpenDoc token.

```
#define INCL_ODNAMERESOLVER
#define INCL_ODAPI
#include <os2.h>

ODOSLToken    *token;
ODDesc        *rv;

rv = GetUserToken(token);
```

---

### GetUserToken Parameter - token

**token** (ODOSLToken \*) - input  
A reference to the OpenDoc token whose descriptor is to be extracted.

---

### GetUserToken Return Value - rv

**rv** (ODDesc \*) - returns  
A reference to a descriptor object contained inside this token.

---

### GetUserToken - Parameters

**token** (ODOSLToken \*) - input  
A reference to the OpenDoc token whose descriptor is to be extracted.

**rv** (ODDesc \*) - returns

A reference to a descriptor object contained inside this token.

---

## GetUserToken - Remarks

This method returns the descriptor that represents your private data. You must not dispose of this descriptor; OpenDoc will dispose of it for you.

The *token* parameter must be a valid OpenDoc token; you can call the [IsODToken](#) method to check whether a token is valid.

---

## GetUserToken - Exception Handling

kODErrInvalidParameter

The specified token is not an OpenDoc token.

---

## GetUserToken - Related Methods

### Related Methods

- [ODNameResolver::IsODToken](#)
- 

## GetUserToken - Topics

### Class:

ODNameResolver

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

## IsODToken

---

## IsODToken - Syntax

This method indicates whether the specified token object has been initialized to the proper format for an OpenDoc token.



```
#define INCL_ODNAMERESOLVER
#define INCL_ODAPI
#include <os2.h>

ODOSLToken    *token;
ODBoolean     rv;

rv = IsODToken(token);
```

-----

## IsODToken Parameter - token

**token** (ODOSLToken \*) - input  
A reference to the token to be checked.

-----

## IsODToken Return Value - rv

**rv** (ODBoolean) - returns  
A flag indicating whether the specified token is a token object created by one of your object accessors.

kODTrue	The token was created by one of your object accessors, and is an OpenDoc token.
kODFalse	The token was not created by one of your object accessors.

-----

## IsODToken - Parameters

**token** (ODOSLToken \*) - input  
A reference to the token to be checked.

**rv** (ODBoolean) - returns  
A flag indicating whether the specified token is a token object created by one of your object accessors.

kODTrue	The token was created by one of your object accessors, and is an OpenDoc token.
kODFalse	The token was not created by one of your object accessors.

-----

## IsODToken - Remarks

You can call this method from the [CallCompareProc](#) method of your part's semantic interface. The *obj1* or *obj2* parameter to that method can be either an [ODDesc](#) object that describes data or an [ODOSLToken](#) object created by one of your object accessors. To distinguish between these two possibilities, you can pass each input parameter in turn to the IsODToken method. If this method returns kODTrue, the input parameter is a token object; if it returns kODFalse, the input parameter is a descriptor object.

---

# IsODToken - Related Methods

## Related Methods

- [ODSemanticInterface::CallCompareProc](#)

---

# IsODToken - Topics

## Class:

[ODNameResolver](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

# Resolve

---

# Resolve - Syntax

This method resolves the object specifier into a token that identifies the target object.

```
#define INCL_ODNAMERESOLVER
#define INCL_ODAPI
#include <os2.h>

ODObjectSpec      *theObject;
ODOSLToken        *token;
ODPart            *contextPart;

Resolve(theObject, token, contextPart);
```

---

# Resolve Parameter - theObject

**theObject** (ODObjectSpec \*) - input

A reference to the object specifier to be resolved.

---

## Resolve Parameter - token

**token** (ODOSLToken \*) - in/out

A reference to the final token produced by the resolution.

---

## Resolve Parameter - contextPart

**contextPart** (ODPart \*) - input

A reference to the part that is calling this method-that is, the context for the supplied token.

---

## Resolve - Return Value

None.

---

## Resolve - Parameters

**theObject** (ODObjectSpec \*) - input

A reference to the object specifier to be resolved.

**token** (ODOSLToken \*) - in/out

A reference to the final token produced by the resolution.

**contextPart** (ODPart \*) - input

A reference to the part that is calling this method-that is, the context for the supplied token.

None.

---

## Resolve - Remarks

OpenDoc calls this method to resolve object specifiers in the *direct* parameter of an OSA event. In addition, your part should call this method to begin the resolution of an object specifier. Note that this method does not allow you to specify any callback flags. These flags are set by the semantic interface of each part using the semantic interface's [SetOSLSupportFlags](#) method. This method may call the object accessors of one or more parts to resolve the object specifier.

The *contextPart* parameter must be a reference to the part that calls this method.

Your part is responsible for deleting the returned token when it is no longer needed by calling the name resolver's [DisposeToken](#) method.

---

## Resolve - Exception Handling

kODOutOfMemory

There was not enough memory to allocate the needed internal structures.

The OSA Event Manager may throw an exception if the specified part was invalid or the part does not support the semantic interface extension.

Parts and OpenDoc may throw other exceptions.

---

## Resolve - Related Methods

### Related Methods

- [ODNameResolver::CallObjectAccessor](#)
  - [ODNameResolver::DisposeToken](#)
  - [ODSemanticInterface::CallObjectAccessor](#)
  - [ODSemanticInterface::SetOSLSupportFlags](#)
- 

## Resolve - Topics

### Class:

[ODNameResolver](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

## ODNameSpace

**Class Definition File:** NAMSPAC.IDL

### Class Hierarchy

SOMObject

ODObject

**ODNameSpace**

### Description

An object of the ODNameSpace class associates an object or data structure with a unique key and provides a fast way to retrieve information.

A name-space object allows a part to identify an object or value using a unique key, which can be passed easily between parts. Parts can use name spaces to store references to parts that support, for example, scripting systems. A part can then iterate over the contents of the name space to obtain a list of scriptable parts or to send a message to all of the scriptable parts. Name spaces can also be written to a storage unit and read back in later.

The ODNameSpace class is an abstract class that defines the basic name-space functionality. OpenDoc defines the subclasses corresponding to two types of name spaces, object and value. An object of the [ODObjectNameSpace](#) class represents an object name space, which stores objects; an object of the [ODValueNameSpace](#) class represents a value name space, which stores values of any type as byte arrays.

Your part creates a new name space by calling the name-space manager's [CreateNameSpace](#) method, specifying a unique name that OpenDoc can use to identify the new name space. Entries within the name space must similarly be identified by a unique key. To obtain a

reference to an existing name space, your part calls the name-space manager's [HasNameSpace](#) method.

Name spaces can be arranged hierarchically to allow you to search multiple name spaces for a single key. Searches move from the child to parent name space to its parent space until the entry is found or until there are no more name spaces to search. A search also stops if the type of the parent name space is different from the type of the child name space, for example, if a value name space is the child of an object name space.

## Methods

The methods defined by the ODNameSpace class include:

- [Exists](#)
- [GetName](#)
- [GetParent](#)
- [GetType](#)
- [ReadFromFile](#)
- [ReadFromStorage](#)
- [SetType](#)
- [Unregister](#)
- [WriteToFile](#)
- [WriteToStorage](#)

## Overridden Methods

There are currently no methods overridden by the ODNameSpace class.

---

# Exists

---

## Exists - Syntax

This method indicates whether this name space contains an entry with the specified key.

```
#define INCL_ODNAMESPACE
#define INCL_ODAPI
#include <os2.h>

ODISOStr    key;
ODBoolean   rv;

rv = Exists(key);
```

---

## Exists Parameter - key

**key** ([ODISOStr](#)) - input  
The key for the requested entry.

---

## Exists Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating the key exists within this name space.

kODTrue

The key exists within this name space.

kODFalse

The key does not exist within this name space.

---

## Exists - Parameters

**key** ([ODISOSTr](#)) - input

The key for the requested entry.

**rv** ([ODBoolean](#)) - returns

A flag indicating the key exists within this name space.

kODTrue

The key exists within this name space.

kODFalse

The key does not exist within this name space.

---

## Exists - Topics

**Class:**

ODNameSpace

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

---

## GetName

---

## GetName - Syntax

This method returns the unique name of this name space.

```
#define INCL_ODNAMESPACE
#define INCL_ODAPI
#include <os2.h>
```

```
ODISOSTr    rv;
```

```
rv = GetName();
```

---

# GetName Return Value - rv

**rv** ([ODISOStr](#)) - returns  
The name of this name space.

---

## GetName - Parameters

**rv** ([ODISOStr](#)) - returns  
The name of this name space.

---

## GetName - Remarks

When you create a name space, you specify its name in the *spaceName* parameter of the name-space manager's [CreateNameSpace](#) method.

When you no longer need the returned name, you should deallocate it.

---

## GetName - Related Methods

### Related Methods

- [ODNameSpaceManager::CreateNameSpace](#)
- 

## GetName - Topics

**Class:**  
[ODNameSpace](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## GetParent

---

# GetParent - Syntax

This method returns a reference to the parent name space of this name space.

```
#define INCL_ODNAMESPACE
#define INCL_ODAPI
#include <os2.h>

ODNameSpace      *rv;

rv = GetParent();
```

-----

## GetParent Return Value - rv

**rv** (ODNameSpace \*) - returns  
A reference to the parent name space or KODNULL if this name space has no parent.

-----

## GetParent - Parameters

**rv** (ODNameSpace \*) - returns  
A reference to the parent name space or KODNULL if this name space has no parent.

-----

## GetParent - Remarks

When you create a name space, you specify its parent name space in the *inheritsFrom* parameter of the name-space manager's [CreateNameSpace](#) method.

-----

## GetParent - Related Methods

### Related Methods

- [ODNameSpaceManager::CreateNameSpace](#)

-----

## GetParent - Topics

**Class:**  
ODNameSpace



Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## GetType

---

## GetType - Syntax

This method returns the type of the name space.

```
#define INCL_ODNAMESPACE  
#define INCL_ODAPI  
#include <os2.h>
```

```
ODNSTypeSpec    rv;
```

```
rv = GetType();
```

---

## GetType Return Value - rv

**rv** ([ODNSTypeSpec](#)) - returns

The type of this name space. This parameter can be set to one of the following values:

kODNSDataTypeODOObject	An object name space.
kODNSDataTypeODValue	An value name space.

---

## GetType - Parameters

**rv** ([ODNSTypeSpec](#)) - returns

The type of this name space. This parameter can be set to one of the following values:

kODNSDataTypeODOObject	An object name space.
kODNSDataTypeODValue	An value name space.

---

## GetType - Remarks

When you create a name space, you specify its type in the *type* parameter of the name-space manager's [CreateNameSpace](#) method.

---

## GetType - Related Methods

### Related Methods

- [ODNameSpace::SetType](#)
  - [ODNameSpaceManager::CreateNameSpace](#)
- 

## GetType - Topics

### Class:

ODNameSpace

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## ReadFromFile

---

## ReadFromFile - Syntax

This method reads the content of this name space from a storage-unit view.

```
#define INCL_ODNAMESPACE
#define INCL_ODAPI
#include <os2.h>

ODByteArray      *file;

ReadFromFile(file);
```

---

## ReadFromFile Parameter - file

**file** ([ODByteArray \\*](#)) - input

A reference to the storage unit from which data is to be read.

---

## ReadFromFile - Return Value

None.

---

## ReadFromFile - Parameters

**file** ([ODByteArray \\*](#)) - input

A reference to the storage unit from which data is to be read.

None.

---

## ReadFromFile - Remarks

You call this method to read in the content of a name space that was written to a storage unit using the [WriteToStorage](#) method.

The content of this name space is stored in the storage unit as a single stream of data. You should focus the storage unit on the appropriate property and value before calling this emthod.

This method does not remove the existing content of this name space, but it does overwrite any entry that has the same key as an entry from the storage unit. After this method executes successfully, this name space's new entries will correspond to entries stored in the storage unit.

---

## ReadFromFile - Exception Handling

kODErrInvalidName

The stored name does not match the name of this name space.

---

## ReadFromFile - Related Methods

### Related Methods

- [ODNameSpace::WriteToStorage](#)
- 

## ReadFromFile - Topics

### Class:

ODNameSpace

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## ReadFromStorage

---

### ReadFromStorage - Syntax

This method reads the content of this name space from the specified storage-unit view.

```
#define INCL_ODNAMESPACE
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitView      *view;

ReadFromStorage (view);
```

---

### ReadFromStorage Parameter - view

**view** (ODStorageUnitView \*) - input  
A reference to the storage-unit view from which data is to be read.

---

### ReadFromStorage - Return Value

None.

---

### ReadFromStorage - Parameters

**view** (ODStorageUnitView \*) - input  
A reference to the storage-unit view from which data is to be read.

None.

---

## ReadFromStorage - Remarks

You call this method to read in the content of a name space that was written to a storage unit using the [WriteToStorage](#) method.

The content of this name space is stored in the storage unit as a single stream of data. The focus context for the specified storage-unit view should be the value from which the name space is to be read.

This method does not remove the existing content of this name space, but it does overwrite any entry that has the same key as an entry from the storage unit. After this method executes successfully, this name space's new entries will correspond to entries read from the storage unit.

---

## ReadFromStorage - Exception Handling

kODErrInvalidName

The stored name does not match the name of this name space.

---

## ReadFromStorage - Related Methods

### Related Methods

- [ODNameSpace::WriteToStorage](#)
- 

## ReadFromStorage - Topics

### Class:

ODNameSpace

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## SetType

---

## SetType - Syntax

This method sets the type of the name space.

```
#define INCL_ODNAMESPACE
#define INCL_ODAPI
#include <os2.h>
```

```
ODNSTypeSpec    type;
```

```
SetType(type);
```

-----

## SetType Parameter - type

**type** (ODNSTypeSpec) - input

The type of this name space. This parameter can be set to one of the following values:

```
kODNSDataTypeODOObject    An object name space.
kODNSDataTypeODValue       An value name space.
```

-----

## SetType - Return Value

None.

-----

## SetType - Parameters

**type** (ODNSTypeSpec) - input

The type of this name space. This parameter can be set to one of the following values:

```
kODNSDataTypeODOObject    An object name space.
kODNSDataTypeODValue       An value name space.
```

None.

-----

## SetType - Remarks

This method is called by subclasses of the [ODNameSpace](#) class to set the type of the name space. Your part should not call this method directly.

-----

## SetType - Related Methods

## Related Methods

- [ODNameSpace::GetType](#)

---

# SetType - Topics

## Class:

ODNameSpace

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

# Unregister

---

## Unregister - Syntax

This method removes the specified entry from this name space.

```
#define INCL_ODNAMESPACE
#define INCL_ODAPI
#include <os2.h>
```

```
ODISOSTr    key;
```

```
Unregister(key);
```

---

## Unregister Parameter - key

**key** ([ODISOSTr](#)) - input

The key for the entry to be removed.

---

## Unregister - Return Value

None.

-----

## Unregister - Parameters

**key** ([ODISOStr](#)) - input  
The key for the entry to be removed.

None.

-----

## Unregister - Remarks

If this name space does not contain an entry with the specified key, this method returns without raising an exception. This method does not search the parent name space for the key.

-----

## Unregister - Related Methods

### Related Methods

- [ODObjectNameSpace::Register](#)
- [ODValueNameSpace::Register](#)

-----

## Unregister - Topics

**Class:**  
[ODNameSpace](#)

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

-----

## WriteToFile

-----

## WriteToFile - Syntax



This method writes the name space out as a stream to the specified file.

```
#define INCL_ODNAMESPACE
#define INCL_ODAPI
#include <os2.h>

ODByteArray      *file;

WriteToFile(file);
```

---

## WriteToFile Parameter - file

**file** ([ODByteArray](#) \*) - input  
The storage unit to which data is to be written.

---

## WriteToFile - Return Value

None.

---

## WriteToFile - Parameters

**file** ([ODByteArray](#) \*) - input  
The storage unit to which data is to be written.

None.

---

## WriteToFile - Remarks

This stream can be read back in by calling the [ReadFromFile](#) method.

---

## WriteToFile - Exception Handling

This method can throw file system errors and storage errors.

---

## WriteToFile - Override Policy

If you subclass [ODNameSpace](#), you should not override this method.

---

## WriteToFile - Related Methods

### Related Methods

- [ODNameSpace::ReadFromFile](#)
- 

## WriteToFile - Topics

### Class:

[ODNameSpace](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

[Related Methods](#)

---

## WriteToStorage

---

## WriteToStorage - Syntax

This method writes the content of this name space to the specified storage-unit view.

```
#define INCL_ODNAMESPACE
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitView    *view;

WriteToStorage(view);
```

---

## WriteToStorage Parameter - view

**view** (ODStorageUnitView \*) - input

A reference to the storage unit in which data is to be written.

---

## WriteToStorage - Return Value

None.

---

## WriteToStorage - Parameters

**view** (ODStorageUnitView \*) - input  
A reference to the storage unit in which data is to be written.

None.

---

## WriteToStorage - Remarks

This method writes the content of this name space as a single stream of data. The focus context for the specified storage-unit view should be the value to which the name space is to be written.

You can read the content of this name space by calling the [ReadFromStorage](#) method.

---

## WriteToStorage - Exception Handling

This method may throw platform-specific exceptions.

---

## WriteToStorage - Related Methods

### Related Methods

- [ODNameSpace::ReadFromStorage](#)
- 

## WriteToStorage - Topics

**Class:**  
ODNameSpace

Select an item:  
[Syntax](#)

[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)  
[Related Methods](#)

---

# ODNameSpaceManager

**Class Definition File:** NMSPCMG.IDL

## Class Hierarchy

SOMObject  
SOMClass  
  ODObject  
    ODBaseNameSpaceManager  
      **ODNameSpaceManager**

## Description

An object of the ODNameSpaceManager class manages the creation and deletion of name spaces among OpenDoc and any associated parts.

When a document is opened, the session object creates a single name-space-manager object. All parts of that document share the name-space manager; you can obtain a reference to it by calling the session object's [GetNameSpaceManager](#) method.

A part creates a name space by calling the name-space manager's [CreateNameSpace](#) method. Name spaces can be arranged hierarchically to facilitate searches among multiple name spaces. To obtain a reference to an existing name space, a part calls the name-space manager's [HasNameSpace](#) method.

For additional information related to name spaces, see the descriptions of the classes [ODNameSpace](#), [ODObjectNameSpace](#), and [ODValueNameSpace](#).

## Methods

The methods defined by the ODNameSpaceManager class include:

- [CreateNameSpace](#)
- [DeleteNameSpace](#)
- [HasNameSpace](#)
- [UpdatePreferences](#)

## Overridden Methods

There are currently no methods overridden by the ODNameSpaceManager class.

---

# CreateNameSpace

---

## CreateNameSpace - Syntax

This method creates a new name space.

```
#define INCL_ODNAMESPACEMANAGER
#define INCL_ODAPI
#include <os2.h>

ODISOSTr      spaceName;
ODNameSpace   *inheritsFrom;
```

```
ODULong      numExpectedEntries;
ODNSTypeSpec type;
ODNameSpace  *rv;

rv = CreateNameSpace(spaceName, inheritsFrom,
                    numExpectedEntries, type);
```

-----

## CreateNameSpace Parameter - spaceName

**spaceName** (ODISOStr) - input  
The unique name for the new name space.

-----

## CreateNameSpace Parameter - inheritsFrom

**inheritsFrom** (ODNameSpace \*) - input  
A reference to the parent of the new name space or kODNULL if this name space has no parent.

-----

## CreateNameSpace Parameter - numExpectedEntries

**numExpectedEntries** (ODULong) - input  
The expected number of entries to be stored in the new name space.

-----

## CreateNameSpace Parameter - type

**type** (ODNSTypeSpec) - input  
The type of the new name space. This parameter can be set to one of the following values:

kODNSDataTypeODObject  
An object name space.  
kODNSDataTypeODValue  
An value name space.

-----

## CreateNameSpace Return Value - rv

**rv** (ODNameSpace \*) - returns  
A reference to the newly created name space or kODNULL if a name space already exists with the specified name.

-----

# CreateNameSpace - Parameters

**spaceName** ([ODISOStr](#)) - input

The unique name for the new name space.

**inheritsFrom** (ODNameSpace \*) - input

A reference to the parent of the new name space or kODNULL if this name space has no parent.

**numExpectedEntries** ([ODULong](#)) - input

The expected number of entries to be stored in the new name space.

**type** ([ODNSTypeSpec](#)) - input

The type of the new name space. This parameter can be set to one of the following values:

kODNSDataTypeODObject

An object name space.

kODNSDataTypeODValue

An value name space.

**rv** (ODNameSpace \*) - returns

A reference to the newly created name space or kODNULL if a name space already exists with the specified name.

---

## CreateNameSpace - Remarks

This method creates either a new object name space or a new value name space, as specified by the *type* parameter. The *spaceName* parameter should specify a unique name for the new name space.

If the *inheritsFrom* parameter is not null, it should specify a parent name space of the same type as the one being created.

Because the current implementation of name spaces uses a hash table, the value you specify in the *numExpectedEntries* parameter should be a prime number to make a hashing algorithm much more efficient. Choose a prime number close to the expected number of entries. The name space can accomodate more entries than you specify (at the expense of performance), but it cannot grow in size.

---

## CreateNameSpace - Related Methods

### Related Methods

- [ODNameSpaceManager::DeleteNameSpace](#)
- [ODNameSpaceManager::HasNameSpace](#)

---

## CreateNameSpace - Topics

### Class:

ODNameSpaceManager

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Related Methods](#)

---

# DeleteNameSpace

---

## DeleteNameSpace - Syntax

This method deletes the specified name space.

```
#define INCL_ODNAMESPACEMANAGER
#define INCL_ODAPI
#include <os2.h>

ODNameSpace      *theNameSpace;

DeleteNameSpace (theNameSpace);
```

---

## DeleteNameSpace Parameter - theNameSpace

**theNameSpace** (ODNameSpace \*) - input  
A reference to the name space to be deleted.

---

## DeleteNameSpace - Return Value

None.

---

## DeleteNameSpace - Parameters

**theNameSpace** (ODNameSpace \*) - input  
A reference to the name space to be deleted.

None.

---

## DeleteNameSpace - Remarks

This method deletes the specified name space by removing it from the name-space manager's internal tables and deallocating the memory

associated with the object.

If the specified name space was a parent to any other name spaces, those name spaces no longer have a parent name space.

---

## DeleteNameSpace - Related Methods

### Related Methods

- [ODNameSpaceManager::CreateNameSpace](#)

---

## DeleteNameSpace - Topics

### Class:

ODNameSpaceManager

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## HasNameSpace

---

## HasNameSpace - Syntax

This method checks whether a specified name space exists and, if so, returns a reference to it.

```
#define INCL_ODNAMESPACEMANAGER
#define INCL_ODAPI
#include <os2.h>

ODISOStr      spaceName;
ODNameSpace   *rv;

rv = HasNameSpace(spaceName);
```

---

## HasNameSpace Parameter - spaceName

**spaceName** ([ODISOStr](#)) - input

The name of the name space to be found.



---

# HasNameSpace Return Value - rv

**rv** (ODNameSpace \*) - returns  
A reference to the name space or kODNULL if the name space does not exist.

---

## HasNameSpace - Parameters

**spaceName** ([ODISOSTr](#)) - input  
The name of the name space to be found.

**rv** (ODNameSpace \*) - returns  
A reference to the name space or kODNULL if the name space does not exist.

---

## HasNameSpace - Remarks

You can call this method to check whether a name space with the specified name exists or to obtain a reference to a specific name space.

When you create a name space, you specify its name in the *spaceName* parameter of the name-space manager's [CreateNameSpace](#) method.

---

## HasNameSpace - Related Methods

### Related Methods

- [ODNameSpaceManager::CreateNameSpace](#)
- 

## HasNameSpace - Topics

**Class:**  
[ODNameSpaceManager](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## UpdatePreferences

---

## UpdatePreferences - Syntax

This method updates the OpenDoc preferences file to reflect changes to the name mappings maintained by OpenDoc.

```
#define INCL_ODNAMESPACEMANAGER
#define INCL_ODAPI
#include <os2.h>
```

```
UpdatePreferences();
```

---

## UpdatePreferences - Return Value

None.

---

## UpdatePreferences - Parameters

None.

---

## UpdatePreferences - Topics

**Class:**  
ODNameSpaceManager

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## ODObject

**Class Definition File:** ODOBJECT.IDL

**Class Hierarchy**  
SOMObject  
  **ODObject**

## Description

The ODObjct class is the superclass of all OpenDoc classes; it defines features that all subclasses support, such as memory recovery and extensions.

The ODObjct class defines a general memory recovery system through the [Purge](#) method. It also defines an extension mechanism that allows subclasses to extend the functionality of objects. Subclasses are responsible for defining the extensions they support. For more information on extensions, see the class description for [ODExtension](#).

You should never instantiate ODObjct. You can create an object of ODObjct subclass by calling the appropriate factory method, if a factory method exists.

## Methods

The methods defined by the ODObjct class include:

- [AcquireExtension](#)
- [HasExtension](#)
- [InitObject](#)
- [IsEqualTo](#)
- [Purge](#)
- [ReleaseExtension](#)
- [SubClassResponsibility](#)

## Overridden Methods

There are currently no methods overridden by the ODObjct class.

---

# AcquireExtension

---

## AcquireExtension - Syntax

This method returns a reference to the specified extension object.

```
#define INCL_ODOJECT
#define INCL_ODAPI
#include <os2.h>

ODType      extensionName;
ODExtension  rv;

rv = AcquireExtension(extensionName);
```

---

## AcquireExtension Parameter - extensionName

**extensionName** ([ODType](#)) - input

The name of extension to be retrieved. This parameter can be set to a part-specific string or one of the following values:

- kODExtSemanticInterface  
An extension to support a semantic interface in your part.
- kODSettingsExtension  
An extension to add parameters to the Properties notebook.

# AcquireExtension Return Value - rv

**rv** (ODEExtension) - returns  
A reference to the specified extension object.

---

## AcquireExtension - Parameters

**extensionName** (ODType) - input  
The name of extension to be retrieved. This parameter can be set to a part-specific string or one of the following values:

- kODExtSemanticInterface  
An extension to support a semantic interface in your part.
- kODSettingsExtension  
An extension to add parameters to the Properties notebook.

**rv** (ODEExtension) - returns  
A reference to the specified extension object.

---

## AcquireExtension - Remarks

Your part call this method to obtain a reference to an extension object with the specified extension type. If the subclass does not support the specified extension, this method raises an extension. The [ODObject](#) class has no inherent extensions of its own and, therefore, always raises and exception.

Your override of this method should increment the reference count of the returned extension object. When you have finished using that extension object, you should call its [Release](#) method.

---

## AcquireExtension - Exception Handling

kODErrUnsupportedExtension	The specified extension is not supported by this object.
----------------------------	--

---

## AcquireExtension - Override Policy

Your subclass of [ODPart](#) can override this method if your part supports extensions. If your part does not support the specified extension, your override method must call its inherited `AcquireExtension` method at the end of your implementation.

---

## AcquireExtension - Related Methods

**Related Methods**

- [ODObject::HasExtension](#)
- [ODObject::ReleaseExtension](#)

---

## AcquireExtension - Topics

### Class:

[ODObject](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## HasExtension

---

## HasExtension - Syntax

This method indicates whether the object supports the specified extension.

```
#define INCL_ODOBJECT
#define INCL_ODAPI
#include <os2.h>

ODType      extensionName;
ODBoolean    rv;

rv = HasExtension(extensionName);
```

---

## HasExtension Parameter - extensionName

**extensionName** ([ODType](#)) - input  
The name of the extension to be checked.

---

## HasExtension Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the object supports the specified extension.

kODTrue

The object supports the specified extension.

kODFalse

The object does not support the specified extension.

-----

## HasExtension - Parameters

**extensionName** ([ODType](#)) - input

The name of the extension to be checked.

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the object supports the specified extension.

kODTrue

The object supports the specified extension.

kODFalse

The object does not support the specified extension.

-----

## HasExtension - Remarks

Your part can call this method before accessing an object's extension.

The [ODObject](#) class has no inherent extensions of its own and always returns kODFalse.

-----

## HasExtension - Override Policy

Your subclass of [ODPart](#) can override this method if your part supports extensions. If your part does not support the specified extension, your override method must call its inherited HasExtension method at the end of your implementation.

-----

## HasExtension - Related Methods

### Related Methods

- [ODObject::AcquireExtension](#)
- [ODObject::ReleaseExtension](#)

-----

## HasExtension - Topics

### Class:

[ODObject](#)

Select an item:

[Syntax](#)

[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Related Methods](#)

---

# InitObject

---

## InitObject - Syntax

This method initializes the object.

```
#define INCL_ODOBJECT  
#define INCL_ODAPI  
#include <os2.h>
```

```
InitObject();
```

---

## InitObject - Return Value

None.

---

## InitObject - Parameters

None.

---

## InitObject - Remarks

Subsequent calls to this method have no affect.

---

## InitObject - Topics

**Class:**

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

# IsEqualTo

---

## IsEqualTo - Syntax

This method indicates whether the specified object is equal to this object.

```
#define INCL_ODOBJECT
#define INCL_ODAPI
#include <os2.h>

ODObject    object;
ODBoolean   rv;

rv = IsEqualTo(object);
```

---

## IsEqualTo Parameter - object

**object** (ODObject) - input

A reference to an object to be compared with this object.

---

## IsEqualTo Return Value - rv

**rv** (ODBoolean) - returns

A flag indicating whether the specified object is equal to this object.

kODTrue	The objects are the same.
kODFalse	The objects are different.

---

## IsEqualTo - Parameters



**object** (ODOObject) - input

A reference to an object to be compared with this object.

**rv** (ODBoolean) - returns

A flag indicating whether the specified object is equal to this object.

kODTrue

The objects are the same.

kODFalse

The objects are different.

-----

## IsEqualTo - Remarks

You should call this method whenever you need to compare objects for equality. You should never use the equal or not equal operators (for example, the == and != operators of the C++ language) to compare references of two objects.

-----

## IsEqualTo - Topics

### Class:

ODOObject

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

-----

## Purge

-----

## Purge - Syntax

This method releases any unneeded memory during low-memory situations.

```
#define INCL_ODOBJECT
#define INCL_ODAPI
#include <os2.h>
```

```
ODSize    size;
ODSize    rv;
```

```
rv = Purge(size);
```

-----

## Purge Parameter - size

**size** ([ODSize](#)) - input  
The number of bytes needed by OpenDoc.

---

## Purge Return Value - rv

**rv** ([ODSize](#)) - returns  
The number of bytes that were released by this object.

---

## Purge - Parameters

**size** ([ODSize](#)) - input  
The number of bytes needed by OpenDoc.

**rv** ([ODSize](#)) - returns  
The number of bytes that were released by this object.

---

## Purge - Remarks

Your part may call this method, but in general, OpenDoc calls this method in low-memory situations to free any caches, noncritical buffers, or objects; you should not allocate memory for this operation.

Because the [ODObject](#) class does not allocate (or deallocate) any memory, it always returns the value 0.

---

## Purge - Override Policy

Every subclass of [ODObject](#) can override this method and should do so if it creates caches and temporary buffers. Your subclass of [ODPart](#) must override this method or risk running out of available memory. Your override method must call its inherited Purge method at some point in your implementation (it does not matter where). You should save the size value returned by the inherited Purge method and then add it to the size value returned by your override method to determine the amount of memory actually released.

---

## Purge - Topics

**Class:**  
[ODObject](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)

---

# ReleaseExtension

---

## ReleaseExtension - Syntax

This method releases the specified extension object.

```
#define INCL_ODOBJECT
#define INCL_ODAPI
#include <os2.h>

ODExtension    extension;

ReleaseExtension(extension);
```

---

## ReleaseExtension Parameter - extension

**extension** (ODExtension) - input  
A reference to the extension to be released.

---

## ReleaseExtension - Return Value

None.

---

## ReleaseExtension - Parameters

**extension** (ODExtension) - input  
A reference to the extension to be released.

None.

---

## ReleaseExtension - Remarks

OpenDoc or an extension object client calls this method to release an extension object that was previously acquired using the [AcquireExtension](#) method. If the extension was not previously acquired by the caller, the ReleaseExtension method raises an exception. After

this method executes successfully, the specified extension object is no longer guaranteed to be valid.

The [ODObject](#) class has no inherent extensions of its own and, therefore, always raises an exception.

---

## ReleaseExtension - Exception Handling

kODErrUnsupportedExtension

This object did not recognize the specified extension object and, therefore, cannot release it.

---

## ReleaseExtension - Override Policy

Your subclass of [ODPart](#) can override this method if your part supports extensions. If your part does not support the specified extension, your override method must call its inherited ReleaseExtension method at the end of your implementation.

---

## ReleaseExtension - Related Methods

### Related Methods

- [ODObject::AcquireExtension](#)
  - [ODObject::HasExtension](#)
- 

## ReleaseExtension - Topics

### Class:

[ODObject](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## SubClassResponsibility

---

## SubClassResponsibility - Syntax

This method raises an exception to indicate that a subclass failed to override a required method.

```
#define INCL_ODOBJECT
#define INCL_ODAPI
#include <os2.h>
```

```
SubClassResponsibility();
```

---

## SubClassResponsibility - Return Value

None.

---

## SubClassResponsibility - Parameters

None.

---

## SubClassResponsibility - Remarks

OpenDoc calls this method at run time to indicate that a subclass that should have overridden a particular method failed to do so.

---

## SubClassResponsibility - Exception Handling

kODErrSubClassResponsibility

The subclass must override this method. The called method should have, but was not, overridden by the subclass of the class that defines this method.

---

## SubClassResponsibility - Topics

### Class:

ODObject

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

---

# ODObjectIterator

**Class Definition File:** OBJECTITR.IDL

## Class Hierarchy

SOMObject  
  ODObject  
    **ODObjectIterator**

## Description

An object of the ODObjectIterator class provides access to the entries of an object name space.

You use an object iterator to apply an operation to all entries of an object name space. For example, you might use an object iterator to compile a complete list of all the part editors that support certain part types.

Your part creates an object iterator object by calling the object name space's [CreateIterator](#) method, which returns a reference to an object iterator object. The iterator performs an unordered traversal of the name space.

While you are using an object iterator, you should not modify or delete the name space. You must postpone adding entries to or removing entries from the name space until after you have deleted the iterator.

For information related to object name spaces, see the class description for [ODObjectNameSpace](#). For more information on accessing objects through iterators, see the chapter on OpenDoc run-time features in the *OpenDoc Programming Guide*.

## Methods

The methods defined by the ODObjectIterator class include:

- [First](#)
- [IsNotComplete](#)
- [Next](#)

## Overridden Methods

There are currently no methods overridden by the ODObjectIterator class.

---

## First

---

## First - Syntax

This method begins the iteration and returns the first entry in the name space.

```
#define INCL_ODOBJECT
#define INCL_ODAPI
#include <os2.h>

ODISOSTr      *key;
ODObject      **object;
ODULong       *objectLength;

First(key, object, objectLength);
```

---

## First Parameter - key

**key** (ODISOStr \*) - output

A pointer an ISO string representing the first key in the first entry in the name space or kODNULL for an empty name space.

-----

## First Parameter - object

**object** (ODOObject \*\*) - output

A reference to the object in the first entry in the name space.

-----

## First Parameter - objectLength

**objectLength** (ODULong \*) - output

The actual size, in bytes, of the specified object.

-----

## First - Return Value

None.

-----

## First - Parameters

**key** (ODISOStr \*) - output

A pointer an ISO string representing the first key in the first entry in the name space or kODNULL for an empty name space.

**object** (ODOObject \*\*) - output

A reference to the object in the first entry in the name space.

**objectLength** (ODULong \*) - output

The actual size, in bytes, of the specified object.

None.

-----

## First - Remarks

Your part must call this method before calling the object iterator's [IsNotComplete](#) method for the first time. This method can be called multiple times; each time, it resets the iteration.

It is your responsibility to deallocate the ISO string when it is no longer needed. You must also delete the key when it is no longer needed.

You do not need to allocate or deallocate any memory for the *object* object.

---

## First - Exception Handling

kODErrIteratorOutOfSync

The name space was modified while the iteration was in progress.

---

## First - Topics

### Class:

ODObjectIterator

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

---

## IsNotComplete

---

## IsNotComplete - Syntax

This method indicates whether the iteration is incomplete.

```
#define INCL_ODOBJECT
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    rv;
```

```
rv = IsNotComplete();
```

---

## IsNotComplete Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the iteration is incomplete.

kODTrue

The iteration is incomplete.

kODFalse

The iteration is complete.



---

## IsNotComplete - Parameters

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the iteration is incomplete.

kODTrue

The iteration is incomplete.

kODFalse

The iteration is complete.

---

## IsNotComplete - Remarks

Your part calls this method to test whether more entries remain in the name space. This method returns kODTrue if the preceding call to the [First](#) or [Next](#) method found an entry. This method returns kODFalse when you have examined all the entries (that is, when the previous call to [First](#) or [Next](#) returned kODNULL). If the name space is empty, this method always returns kODFalse.

---

## IsNotComplete - Exception Handling

kODErrIteratorNotInitialized

This method was called before calling either the [First](#) or [Next](#) method to begin the iteration.

kODErrIteratorOutOfSync

The name space was modified while the iteration was in progress.

---

## IsNotComplete - Topics

### Class:

[ODObjectIterator](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

---

## Next

---

## Next - Syntax

This method returns the next entry in the name space.

```
#define INCL_ODOBJECT
#define INCL_ODAPI
#include <os2.h>

ODISOSTr *key;
ODObject **object;
ODULong *objectLength;

Next(key, object, objectLength);
```

-----

## Next Parameter - key

**key** (ODISOSTr \*) - output

A pointer to an ISO string representing the key in the next entry in the name space or KODNULL if you have reached the end of the name space.

-----

## Next Parameter - object

**object** (ODObject \*\*) - output

A reference to the object in the next entry in the name space.

-----

## Next Parameter - objectLength

**objectLength** (ODULong \*) - output

The actual size, in bytes, of the specified object.

-----

## Next - Return Value

None.

-----

## Next - Parameters

**key** (ODISOSTr \*) - output

A pointer to an ISO string representing the key in the next entry in the name space or KODNULL if you have reached the end of the name space.

**object** (ODOBJECT \*\*) - output  
A reference to the object in the next entry in the name space.

**objectLength** (ODULong \*) - output  
The actual size, in bytes, of the specified object.

None.

---

## Next - Remarks

If your part calls this method before calling this object iterator's [First](#) method to begin the iteration, then this method works the same as calling the [First](#) method.

It is your responsibility to deallocate the ISO string when it is no longer needed. You must also delete the key when it is no longer needed. You do not need to allocate or deallocate any memory for the *object* parameter.

---

## Next - Exception Handling

kODerrIteratorOutOfSync

The collection was modified while the iteration was in progress.

---

## Next - Topics

**Class:**  
ODOBJECTIterator

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)

---

## ODOBJECTNameSpace

**Class Definition File:** OBJECTNS.IDL

**Class Hierarchy**  
SOMObject  
ODOBJECT  
ODNameSpace  
**ODOBJECTNameSpace**

### Description

An object of the ODOBJECTNameSpace class is a collection of objects, each of which has a unique key to identify the object within the collection.

An object name space allows a part to identify an object using a unique key, which can be passed easily between parts. A part can create an object name space to store a reference to a part object to facilitate communications with its embedded parts.

Your part can create objects of the ODOBJECTNAME\_SPACE class by calling the name-space manager's [CreateNameSpace](#) method, passing the constant KODNSDataTypeODOBJECT for the type of the name space.

Object name spaces can be arranged hierarchially to allow you to search multiple object name spaces for a single key. Searches move from a child object name space to its parent object name space until the entry is found or until there are no more objects name spaces to search.

## Methods

The methods defined by the ODOBJECTNAME\_SPACE class include:

- [CreateIterator](#)
- [GetEntry](#)
- [Register](#)

## Overridden Methods

There are currently no methods overridden by the ODOBJECTNAME\_SPACE class.

---

# CreateIterator

---

## CreateIterator - Syntax

This method creates an object iterator for the entries in this object name space.

```
#define INCL_ODOBJECTNAME_SPACE
#define INCL_ODAPI
#include <os2.h>
```

```
ODOBJECTIterator      *rv;
```

```
rv = CreateIterator();
```

---

## CreateIterator Return Value - rv

**rv** (ODOBJECTIterator \*) - returns

A reference to an object iterator for inheriting over the entries in this object name space.

---

## CreateIterator - Parameters

**rv** (ODOBJECTIterator \*) - returns

A reference to an object iterator for inheriting over the entries in this object name space.

---

## CreateIterator - Remarks

You call this method when you need to apply an operation to all the entries in this name space.

While you are using the returned object iterator, you must not modify this object name space; in particular, you must not register or unergister entries, and you must not delete this object name space.

You must delete the iterator when it is no longer needed.

-----

## CreateIterator - Topics

### Class:

ODObjectNameSpace

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

-----

## GetEntry

-----

## GetEntry - Syntax

This method searches for an entry with the specified key and, if it exists, gets the object associated with that key.

```
#define INCL_ODOBJECTNAMESPACE
#define INCL_ODAPI
#include <os2.h>
```

```
ODISOSTr      key;
ODObject      **object;
ODBoolean     rv;
```

```
rv = GetEntry(key, object);
```

-----

## GetEntry Parameter - key

**key** ([ODISOSTr](#)) - input

The key to be searched for in this object name space.

-----

## GetEntry Parameter - object

**object** (ODOObject \*\*) - output

A reference to the object corresponding to the specified key or KODNULL if the entry is not found.

---

## GetEntry Return Value - rv

**rv** (ODBoolean) - returns

A flag indicating whether the entry is found.

kODTrue

The entry was found.

kODFalse

The entry was not found.

---

## GetEntry - Parameters

**key** (ODISOStr) - input

The key to be searched for in this object name space.

**object** (ODOObject \*\*) - output

A reference to the object corresponding to the specified key or KODNULL if the entry is not found.

**rv** (ODBoolean) - returns

A flag indicating whether the entry is found.

kODTrue

The entry was found.

kODFalse

The entry was not found.

---

## GetEntry - Remarks

If the specified key is found, this method sets the *object* output parameter to a reference to the corresponding object. If the specified key is not found in this name space, this method searches the parent name space. Searches proceed from each object name space to its parent until one of the following happens:

- The entry is found.
- There is no parent name space to search.
- The parent name space is a value name space instead of an object name space.

If the object corresponding to the specified key is a reference-counted object, this method does not increment the object's reference count. For that reason, if you cache that object, you should call its [Acquire](#) method increment its reference count and then call its [Release](#) method when you are finished using the object.

---

## GetEntry - Related Methods

### Related Methods

- [ODNameSpace::Exists](#)
- [ODOObjectNameSpace::Register](#)

---

# GetEntry - Topics

**Class:**  
ODObjectNamespace

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## Register

---

## Register - Syntax

This method adds a new entry to this object name space.

```
#define INCL_ODOBJECTNAMESPACE
#define INCL_ODAPI
#include <os2.h>
```

```
ODISOStr      key;
ODObject      *object;
```

```
Register(key, object);
```

---

## Register Parameter - key

**key** ([ODISOStr](#)) - input  
The key for the new entry.

---

## Register Parameter - object

**object** (ODObject \*) - input  
A reference to the object to be associated with the key.

---

# Register - Return Value

None.

---

## Register - Parameters

**key** ([ODISOSTr](#)) - input  
The key for the new entry.

**object** ([ODOObject \\*](#)) - input  
A reference to the object to be associated with the key.

None.

---

## Register - Remarks

This method stores a reference to the *object* parameter in a new entry in the table. If the specified key already exists in this name space, this method overwrites the old entry with the new one.

When you register a reference-counted object with an object registry, this method does not increment its reference count.

---

## Register - Related Methods

### Related Methods

- [ODNameSpace::Unregister](#)
- 

## Register - Topics

**Class:**  
[ODOObjectNameSpace](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## ODOObjectSpec



**Class Definition File:** ODOBJSPC.IDL

### Class Hierarchy

SOMObject  
  ODObject  
    ODDesc  
      ODObjectSpec

### Description

An object of the ODOObjectSpec class is a wrapper for an *object specifier structure* (type typeObjectSpecifier), a descriptor structure on the OS/2 platform that describes the location of one or more OSA event objects.

If your application creates and sends OSA events that require a target application to locate OSA event objects, your application must create object specifiers for those events.

For more information on OSA events and the typeObjectSpecifier type, see the chapter on responding to OSA events in the *Open Scripting Architecture Guide and Reference for OS/2* . For general information on scripting support in OpenDoc, see the chapter on semantic events and scripting in the *OpenDoc Programming Guide* .

### Methods

The methods defined by the ODOObjectSpec class include:

- [InitODOObjectSpec](#)

### Overridden Methods

There are currently no methods overridden by the ODOObjectSpec class.

---

## InitODOObjectSpec

---

## InitODOObjectSpec - Syntax

This method initializes this object specifier.

```
#define INCL_ODOOBJECTSPEC
#define INCL_ODAPI
#include <os2.h>
```

```
InitODOObjectSpec();
```

---

## InitODOObjectSpec - Return Value

None.

---

## InitODOObjectSpec - Parameters

None.

---

## InitODObjectSpec - Remarks

There is no factory method for the [ODObjectSpec](#) class; after creating a new object specifier, OpenDoc or your part must call this method to initialize the new object specifier.

---

## InitODObjectSpec - Topics

### Class:

ODObjectSpec

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

---

## ODOSAEvent

**Class Definition File:** ODOSAEVT.IDL

### Class Hierarchy

```
SOMObject
  ODOObject
    ODDDesc
      ODDDescList
        ODRRecord
          ODOSAEvent
```

### Description

An object of the ODOSAEvent class is a wrapper for an OSA event structure (type [OSAEvent](#)).

An OSA event structure is a data structure of type [OSAEvent](#) containing a list of keyword-specified descriptor records that name the attributes and parameters of an event. An OSA event structure is different from an OSA event object.

For more information on OSA event structures and the [OSAEvent](#) type, see the chapter on introducing OSA events in the *Open Scripting Architecture Guide and Reference for OS/2*. For general information on scripting support in OpenDoc, see the chapter on semantic events and scripting in the *OpenDoc Programming Guide*.

### Methods

The methods defined by the ODOSAEvent class include:

- [InitODOSAEvent](#)

### Overridden Methods

There are currently no methods overridden by the ODOSAEvent class.

---

## InitODOSAEvent (OS/2)

---

## InitODOSAEvent (OS/2) - Syntax

This method initializes this OSA event structure.

```
#define INCL_ODOSAEvent
#define INCL_ODAPI
#include <os2.h>

InitODOSAEvent();
```

---

## InitODOSAEvent (OS/2) - Return Value

None.

---

## InitODOSAEvent (OS/2) - Parameters

None.

---

## InitODOSAEvent (OS/2) - Remarks

There is no factory method for the [ODOSAEvent](#) class; after creating a new OSA event structure, OpenDoc or your part must call this method to initialize the new OSA event structure.

---

## InitODOSAEvent (OS/2) - Topics

**Class:**  
ODOSAEvent

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

# ODOSLToken

**Class Definition File:** ODOSLTKN.IDL

## Class Hierarchy

```
SOMObject
  ODOObject
    ODDesc
      ODOSLToken
```

## Description

An object of the ODOSLToken class is a wrapper for an OpenDoc token.

OpenDoc owns the format of the data stored in the ODOSLToken class, including information about where the token came from and a reference to a descriptor object.

For general information on tokens and scripting support in OpenDoc, see the chapter on semantic events and scripting in the *OpenDoc Programming Guide* .

## Methods

The methods defined by the ODOSLToken class include:

- [DuplicateODOSLToken](#)
- [InitODOSLToken](#)

## Overridden Methods

There are currently no methods overridden by the ODOSLToken class.

---

## DuplicateODOSLToken

---

## DuplicateODOSLToken - Syntax

This method copies all context information of this OpenDoc token and creates a new OpenDoc token that contains a reference to a descriptor object.

```
#define INCL_ODOSLTKN
#define INCL_ODAPI
#include <os2.h>

ODOSLToken      *rv;

rv = DuplicateODOSLToken();
```

---

## DuplicateODOSLToken Return Value - rv

**rv** (ODOSLToken \*) - returns  
A reference to the new OpenDoc token.

---

## DuplicateODOSLToken - Parameters

**rv** (ODOSLToken \*) - returns  
A reference to the new OpenDoc token.

---

## DuplicateODOSLToken - Remarks

You must call this method whenever you need to copy an OpenDoc token; it ensures that private data in this OpenDoc token gets copied correctly. You might use this method during object resolution when you are referring to a group of objects and you want to make OpenDoc tokens representing each object.

Although technically there is no factory method for the [ODOSLToken](#) class, the DuplicateODOSLToken method could be considered a factory method because it is really the only valid way for you to create a new OpenDoc token.

---

## DuplicateODOSLToken - Exception Handling

kODErrOutOfMemory	There is not enough memory to allocate the OpenDoc token.
-------------------	---

---

## DuplicateODOSLToken - Topics

**Class:**  
ODOSLToken

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Exception Handling](#)

---

## InitODOSLToken

---

## InitODOSLToken - Syntax

This method initializes this OpenDoc token.

```
#define INCL_ODOSLTOKEN
```

```
#define INCL_ODAPI
#include <os2.h>
```

```
InitODOSLToken();
```

---

## InitODOSLToken - Return Value

None.

---

## InitODOSLToken - Parameters

None.

---

## InitODOSLToken - Remarks

There is no factory method for the [ODOSLToken](#) class; after creating a new OpenDoc token, OpenDoc or part must call this method to initialize the new object specifier.

---

## InitODOSLToken - Topics

### Class:

ODOSLToken

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

---

## ODPart

**Class Definition File:** PART.IDL

### Class Hierarchy

SOMObject

ODObject

ODRefCntObject

ODPersistentObject

**ODPart**

## Description

An object of a subclass of `ODPart` represents a part editor.

Parts are the building blocks of OpenDoc documents. A compound document can contain one or more parts, each manipulated by a part editor. A *part editor* is much like an application; it can display and change the content of a part, handle events, perform disk I/O, and accept scripting commands. A part editor directly manipulates the content of its part, called its *intrinsic content*.

The `ODPart` class is an abstract superclass that you must subclass to create your part editor. Your implementation is the executable code that provides the structure and behavior of your parts. If you subclass `ODPart`, you need to override only those methods that represent specific capabilities supported by your part editor.

OpenDoc or another part creates your part object by calling the draft's [CreatePart](#) method. OpenDoc or another part can also access a previously created, stored part by calling the draft's [AcquirePart](#) method. These methods return a reference to a part object.

## Layout and Imaging

OpenDoc provides an environment for managing the geometric relationships among frames and objects when your part draws itself. OpenDoc notifies each part when the part editor must draw the part. Each part is responsible for drawing its own content (including, at certain times, the borders of its embedded frames). A part does not draw the interiors of its embedded frames because they contain the content of other parts (which must draw themselves). Drawing may occur asynchronously, in response to update events, or when activation or deactivation affects highlighting. A part editor typically draws the context of a particular facet. The part editor gets the clipping, transform, and layout information from the facet and its frame, and then must make platform-specific graphics calls to actually draw the lines, polygons, text, and so on, that make up its part.

A container part can define a set of *container properties* (such as text font, background color, or pen width) that it prefers its embedded parts to adopt to enhance the visual continuity between them. Embedded parts that can read a given container part's container properties can then adopt those properties that are appropriate.

## View Type and Presentation

Each frame has a *view type*, the basic representation of a part in that frame. Parts must support the standard set of view types (large icon, small icon, thumbnail and frame view). The *presentation* of a frame describes a particular style of display for a part's content within the frame when its view type is framed. Presentations are part-defined; your part editor determines what types of presentations your part can handle. Your part is notified if its containing part changes the view type or presentation of one of its display frames.

## Frame Negotiation

Each part in an OpenDoc document controls the position, size, and shape of its embedded frames. If an embedded part needs to change the size, shape, or number of frames it is displayed in, it must negotiate for that change with its containing part. Frame negotiation allows an embedded part to communicate its needs to its containing part.

## Initialization and Storage

Your part editor must be able to initialize its parts, as well as read and write their content to and from persistent storage in its document. The data of each part in a document is kept in at least one *storage unit*, the basic unit of persistent storage, distinct from data in other parts. You use the *contents property* of your storage unit for storing all your part's data.

Your part-editor code should never pass out a pointer to `somSelf`, except to support extensions and your embedded-frames iterator. Instead, your part must store and pass out a reference to a *part wrapper*, an object that forwards all method calls to your part. Part wrappers insulate OpenDoc from requiring a direct reference to your part. A reference to a stored wrapper is specified in your part's [InitPart](#) or [InitPartFromStorage](#) method. The reference is defined for the lifetime of your part; once set, it cannot be changed.

## Event Handling

OpenDoc handles events such as mouse clicks and keystrokes, menu commands, activation and deactivation of windows, and other platform-specific events. Part editors do not receive events directly from the operating system; rather, OpenDoc receives events, interprets them, and dispatches them, using the dispatcher, to the appropriate part editor's [HandleEvent](#) method. Based on information in an event, your part editor might update your part, open or close windows, perform editing operations, transfer data, perform menu commands, or perform other operations.

For more information on event handling in OpenDoc, see the chapter on user events in the *OpenDoc Programming Guide*.

## Keeping Track of Display Frames and Embedded Frames

Your part can display its content in one or more frames at the same time, displaying either duplicate views or different aspects of the same part. This makes it easy for your part to be visible in several windows or to have several different presentations. Identical views of your part in two or more separate display frames must be *synchronized*, meaning that any editing or other changes to one display frame forces updating of the others. If your part is a containing part, your part calls its embedded frame's [AttachSourceFrame](#) method to initiate synchronization of its embedded frames.

Your part must maintain a list of all frames in which it is displayed. In addition, if your part is a containing part, it must maintain a list of all frames directly embedded within it. For embedded frames, you may use whatever format for that list that works best for your implementation, but you must provide an embedded-frames iterator (a subclass of [ODEmbeddedFramesIterator](#)) to allow access to the frames.

For efficient memory usage, your part does not need to keep frame objects in memory for all its embedded frames or display frames. Instead,

it can release frames as they become invisible through scrolling and then get them back from the draft as they become visible.

For more information related to frame objects, see the class description for [ODFrame](#).

## Part Kinds and Binding

Parts store their data in formats defined by a *part kind*, a typing scheme analogous to a file type. OpenDoc uses the concept of part kinds to determine what part editor is to be associated with a given part in a document. You can design your part editor to manipulate more than one part kind, and your part can be stored with multiple representations of its data, each of a different part kind. To maximize fidelity in case your part is later edited by a different part editor, your part editor must always store its part kinds in order from highest fidelity to lowest.

## Linking

Your part uses linking to export updatable data to another part, or to incorporate or embed an updatable copy of another part's data into your part. If your part is the source of a link, you interact with a link-source object; if your part is the destination of a link, you interact with a link object.

## Overriding ODPart Methods

The following table lists the methods of the ODPart class that you must override to have a functioning part editor, as well as those that you can optionally override to participate in specific protocols. Note that some protocols, such as layout, imaging, and activation, are required of all part editors, and you must override some or all of the methods associated with them. Other protocols, such as embedding or undo, are not required, and you need not override any of their methods if your parts do not participate. It is, of course, strongly recommended that your parts participate in all protocols.

Protocol	Required Overrides	Optional Overrides
Layout	<a href="#">AttachSourceFrame</a> <a href="#">ContainingPartPropertiesUpdated</a> <a href="#">DisplayFrameAdded</a> <a href="#">DisplayFrameClosed</a> <a href="#">DisplayFrameConnected</a> <a href="#">DisplayFrameRemoved</a> <a href="#">FacetAdded</a> <a href="#">FacetRemoved</a> <a href="#">FrameShapeChanged</a> <a href="#">GeometryChanged</a> <a href="#">Open</a> <a href="#">SequenceChanged</a>	<a href="#">AcquireContainingPartProperties*</a> <a href="#">RevealFrame*</a>
Imaging	<a href="#">CanvasChanged</a> <a href="#">Draw</a> <a href="#">GetPrintResolution</a> <a href="#">HighlightChanged</a> <a href="#">PresentationChanged</a> <a href="#">ViewTypeChanged</a>	<a href="#">AdjustBorderShape*</a> <a href="#">CanvasUpdated*</a>
Activation	<a href="#">AbortRelinquishFocus</a> <a href="#">BeginRelinquishFocus</a> <a href="#">CommitRelinquishFocus</a> <a href="#">FocusAcquired</a> <a href="#">FocusLost</a>	
User Events	<a href="#">AdjustMenus</a> <a href="#">HandleEvent</a>	
Storage	<a href="#">CloneInto**</a> <a href="#">ClonePartInfo</a> <a href="#">Externalize**</a> <a href="#">ExternalizeKinds</a> <a href="#">InitPart</a> <a href="#">InitPartFromStorage</a> <a href="#">ReadPartInfo</a> <a href="#">WritePartInfo</a>	<a href="#">somInit**</a> <a href="#">somUninit**</a>
Binding	<a href="#">ChangeKind</a>	
Memory Management	<a href="#">ReleaseAll**</a>	<a href="#">Acquire**</a> <a href="#">Purge**</a> <a href="#">Release**</a>
Linking	<a href="#">LinkStatusChanged</a>	<a href="#">CreateLink</a> <a href="#">EditInLinkAttempted*</a> <a href="#">FulfillPromise</a> <a href="#">LinkUpdated</a> <a href="#">RevealLink</a>
Embedding		<a href="#">CreateEmbeddedFramesIterator*</a> <a href="#">EmbeddedFrameUpdated*</a>



	<code>RemoveEmbeddedFrame*</code>
	<code>RequestEmbeddedFrame*</code>
	<code>RequestFrameShape*</code>
	<code>UsedShapeChanged*</code>
Clipboard	<code>FulfillPromise</code>
Drag and Drop	<code>DragEnter</code>
	<code>DragLeave</code>
	<code>DragWithin</code>
	<code>Drop</code>
	<code>DropCompleted</code>
	<code>FulfillPromise</code>
Undo	<code>DisposeActionState</code>
	<code>ReadActionState</code>
	<code>RedoAction</code>
	<code>UndoAction</code>
	<code>WriteActionState</code>
Extensions	<code>AcquireExtension**</code>
	<code>HasExtension**</code>
	<code>ReleaseExtension**</code>
Semantic Events	<code>EmbeddedFrameSpec*</code>

#### Note:

\* Required of all parts that support embedding.

\*\* Defined in a superclass of ODPart; see "Overriding Inherited Methods."

### Overriding Inherited Methods

The following methods are inherited and available for use by your subclass of ODPart. You need to override only those methods that represent specific capabilities supported by your part editor.

#### somInit

This method initializes the instance variables in a SOM object; it is inherited from the SOMObject class.

If you subclass ODPart, you can override this method. Your override method does not need to call its inherited method; the inherited method is automatically called for you by the SOM library.

Your override of this method should initialize the new instance variables in your part. The SOM library calls this method when your part is created. You must not do anything that might cause this method to fail. This limits you to operations such as setting pointer variables to null, setting numeric variables to appropriate values, and making similar assignments from constants. If you have any initialization code that can potentially fail, it must be handled in your part's [InitPart](#) or [InitPartFromStorage](#) method.

#### somUninit

This method disposes of the storage created for a SOM object; it is inherited from the SOMObject class.

If you subclass ODPart, you can override this method. Your override method does not need to call its inherited method; the inherited method is automatically called for you by the SOM library.

Your override of this method should dispose of any storage created for your part, including any storage related to additional instances variables initialized in your part. The SOM library calls this method when your part is deleted; this method must not fail.

#### HasExtension

This method indicates whether an object supports a specified extension; it is inherited from the [ODObject](#) class.

```
ODBoolean HasExtension (ODType extensionName);
```

If you subclass ODPart, you can override this method if your part supports extensions. If your part does not support the specified extension, your override method must call its inherited method at the end of your implementation. The client of the extension object must be prepared in the event that the specified extension is not supported.

Your override of this method should return kODTrue if your part supports the specified extension; otherwise, it should return kODFalse. A client of an extension object calls this method to determine whether your part supports extensions, such as scripting.

#### AcquireExtension

This method returns a reference to a specified extension object; it is inherited from the [ODObject](#) class.

```
ODExtension AcquireExtension (ODType extensionName);
```

If you subclass `ODPart`, you can override this method if your part supports extensions. If your part does not support the specified extension, your override method must call its inherited method at the end of your implementation. The client of an extension object must be prepared in the event that the specified extension is not supported.

Your override of this method should return a reference to your part's extension object (for example, a semantic-interface object). A client of an extension object calls this method to obtain a reference to the specified extension.

### **ReleaseExtension**

This method releases a specified extension object; it is inherited from the `ODObject` class.

```
void ReleaseExtension (ODExtension extension);
```

If you subclass `ODPart`, you can override this method if your part supports extensions. If your part does not support the specified extension, your override method must call its inherited method at the end of your implementation. The client of the extension object must be prepared in the event that the specified extension is not supported.

Your override of this method should release the specified extension object (acquired when your part's `AcquireExtension` method was called). A client of an extension object calls this method when it finishes working with your part's extension.

### **Acquire**

This method increments an object's reference count by 1; it is inherited from the `ODRefCountObject` class.

```
void Acquire ();
```

If you subclass `ODPart`, you can override this method if your part performs any specific actions when its reference count is incremented. Your override method must call its inherited method at the beginning of your implementation.

The inherited `Acquire` method increments your part's reference count by 1. A part editor calls this method when it stores a reference to your part. When the references is replaced or returned, a part editor calls your part's `Release` method.

### **Release**

This method decrements an object's reference count by 1 and tells the draft to release the object from memory if the object's reference count becomes 0; it is inherited from the `ODRefCountObject` class.

```
void Release ();
```

If you subclass `ODPart`, you can override this method if your part needs to release an object and reclaim valuable resources, such as memory. Your override method must call its inherited method at the beginning of your implementation.

The inherited `Release` method decrements your part's reference count by 1. The inherited method may delete your part from memory (if your part's reference count becomes 0). A part editor calls this method when it no longer needs a reference to your part.

### **ReleaseAll**

This method releases all transitory references to other reference-counted objects; it is inherited from the `ODPersistentObject` class.

```
void ReleaseAll ();
```

If you subclass `ODPart`, you can override this method if your part maintains references to other persistent objects. Your override method must call its inherited method at the end of your implementation.

Your override of this method should release all persistent objects it points to, remove any link specifications your part has written to the clipboard, relinquish all foci, and clear any undo actions it has written to the undo stack. For any extension that cannot be deleted because of a positive reference count, your override method must also call its base extension's `BaseRemoved` method. If your part's embedded frames are being traversed by an embedded-frames iterator when your part is deleted, your override method must also call your embedded-frames iterator's `PartRemoved` method.

OpenDoc calls this method once before your part is deleted, for example, when your part's document is closed. If the part being deleted has a promise, the promise is resolved before the part is deleted.

### CloneInto

This method clones a persistent object by copying its data into a specified storage unit; it is inherited from the [ODPersistentObject](#) class.

```
void CloneInto (ODDraftKey key, ODStorageUnit toSU, ODFrame scope);
```

If you subclass [ODPart](#), you must override this method to support data interchange of internal data. Your override method must call its inherited method at the beginning of your implementation. The inherited method clones the [kODPropCreateDate](#), [kODPropModDate](#), and [kODPropModUser](#) properties.

Your override of this method should write your part's data into the specified storage unit and clone any additional objects to which this part has strong and weak persistent references and that are within the specified scope. OpenDoc calls this method if your part is being copied to the clipboard, the drag-and-drop object, or a link source object.

### Externalize

This method writes a persistent object to storage; it is inherited from the [ODPersistentObject](#) class.

```
void Externalize ();
```

If you subclass [ODPart](#), you can override this method if your part maintains persistent content. Your override method must call its inherited method at the beginning of your implementation.

Your override of this method should write your part's persistent state to the properties and values your part created when it initialized itself in your part's [InitPart](#) method; it should write sufficient information to allow your [InitPartFromStorage](#) method to restore your part to its current state. Your part editor calls this method when your part stores its content data.

### Purge

This method frees memory on request; it is inherited from the [ODObject](#) class.

```
ODSize Purge (ODSize size);
```

If you subclass [ODObject](#), you can override this method and should do so if it creates caches and temporary buffers. If you subclass [ODPart](#), you must override this method or risk running out of available memory. Your override method must call its inherited method at some point in your implementation (it does not matter where). You should save the size value returned by the inherited method because you will need to compute the value returned from your override method.

Your override of this method should free any caches, noncritical buffers, or objects (up to the amount of memory specified). If absolutely necessary, your part can first write any data it needs to persistent storage, although that is not recommended because writing to storage requires additional memory allocation. Your override of this method should add the number of bytes actually freed to the number returned by the inherited method and return the sum as the total amount of memory released. OpenDoc calls this method in low-memory situations; you should not allocate memory for this operation.

### Methods

The methods defined by the [ODPart](#) class include the following, grouped by purpose:

#### Initialization and Storage

- [ClonePartInfo](#)
- [ExternalizeKinds](#)
- [InitPart](#)
- [InitPartFromStorage](#)
- [ReadPartInfo](#)
- [WritePartInfo](#)

#### Binding

- [ChangeKind](#)
- [GetRealPart](#)
- [IsRealPart](#)
- [ReleaseRealPart](#)

## Imaging

- [CanvasChanged](#)
- [CanvasUpdated](#)
- [Draw](#)
- [GetPrintResolution](#)
- [HighlightChanged](#)

## Facet Manipulation

- [FacetAdded](#)
- [FacetRemoved](#)
- [GeometryChanged](#)

## Frame Negotiation

- [FrameShapeChanged](#)
- [RequestEmbeddedFrame](#)
- [RequestFrameShape](#)

## Display Frame Manipulation

- [AttachSourceFrame](#)
- [DisplayFrameAdded](#)
- [DisplayFrameClosed](#)
- [DisplayFrameConnected](#)
- [DisplayFrameRemoved](#)
- [PresentationChanged](#)
- [Open](#)
- [SequenceChanged](#)
- [ViewTypeChanged](#)

## Embedded Frame Manipulation

- [AdjustBorderShape](#)
- [CreateEmbeddedFramesIterator](#)
- [EmbeddedFrameSpec](#)
- [RemoveEmbeddedFrame](#)
- [RevealFrame](#)
- [UsedShapeChanged](#)

## Focus Manipulation

- [AbortRelinquishFocus](#)
- [BeginRelinquishFocus](#)
- [CommitRelinquishFocus](#)
- [FocusAcquired](#)
- [FocusLost](#)

## Event Handling

- [AdjustMenus](#)
- [HandleEvent](#)

## Undo

- [DisposeActionState](#)
- [ReadActionState](#)
- [RedoAction](#)
- [UndoAction](#)
- [WriteActionState](#)

## Linking

- [BreakLink](#)
- [BreakLinkSource](#)
- [CreateLink](#)
- [EditInLinkAttempted](#)
- [EmbeddedFrameUpdated](#)
- [LinkBroken](#)
- [LinkStatusChanged](#)
- [LinkUpdated](#)
- [RevealLink](#)
- [ShowLink](#)
- [UpdateFromLinkSource](#)

## Data Interchange

- [DragEnter](#)
- [DragLeave](#)
- [DragWithin](#)
- [Drop](#)
- [DropCompleted](#)
- [FulfillPromise](#)

## Container Properties

- [AcquireContainingPartProperties](#)
- [ContainingPartPropertiesUpdated](#)

## Manu Bar Creation

- [CreateRootMenuBar](#)

## Overridden Methods

There are currently no methods overridden by the ODPart class.

# AbortRelinquishFocus

## AbortRelinquishFocus - Syntax

This method should reestablish this part's ownership of the focus specified in a previous call to this part's [BeginRelinquishFocus](#) method.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODTypeToken    focus;
ODFrame        *ownerFrame;
ODFrame        *proposedFrame;

AbortRelinquishFocus(focus, ownerFrame, proposedFrame);
```

## AbortRelinquishFocus Parameter - focus

**focus** ([ODTypeToken](#)) - input

A tokenized string representing the focus type that was to be relinquished.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

- [kODClipboardFocus](#)  
The frame with the clipboard focus has access to the clipboard.
- [kODKeyFocus](#)  
The frame with keyboard focus receives keyboard events (excluding page-movement key events).
- [kODMenuFocus](#)  
The frame with menu focus receives menu events.
- [kODModalFocus](#)  
A frame with modal focus is notifying other frames that it is the only currently modal frame.
- [kODMouseFocus](#)

The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

**kODScrollingFocus** The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

**kODSelectionFocus** The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

-----

## AbortRelinquishFocus Parameter - ownerFrame

**ownerFrame** (ODFrame \*) - input  
A reference to the display frame that currently possesses the focus.

-----

## AbortRelinquishFocus Parameter - proposedFrame

**proposedFrame** (ODFrame \*) - input  
A reference to a frame that originally requested the focus.

-----

## AbortRelinquishFocus - Return Value

None.

-----

## AbortRelinquishFocus - Parameters

**focus** (ODTypeToken) - input  
A tokenized string representing the focus type that was to be relinquished.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

**kODClipboardFocus** The frame with the clipboard focus has access to the clipboard.

**kODKeyFocus** The frame with keyboard focus receives keyboard events (excluding page-movement key events).

**kODMenuFocus** The frame with menu focus receives menu events.

**kODModalFocus** A frame with modal focus is notifying other frames that it is the only currently modal frame.

**kODMouseFocus** The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

**kODScrollingFocus** The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

**kODSelectionFocus**

The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

**ownerFrame** (ODFrame \*) - input

A reference to the display frame that currently possesses the focus.

**proposedFrame** (ODFrame \*) - input

A reference to a frame that originally requested the focus.

None.

-----

## AbortRelinquishFocus - Remarks

A request for a focus set is conditional; if any focus owner is unwilling to relinquish a focus, then none are acquired. Your part's `AbortRelinquishFocus` method should give those focus owners who have indicated willingness to relinquish the focus an opportunity to back out of changes initiated when OpenDoc first called your part's `BeginRelinquishFocus` method. When a request for a focus set fails because one of the focus owners will not relinquish ownership, OpenDoc aborts the request to relinquish all of the foci by calling each part's `AbortRelinquishFocus` method.

Your part's `AbortRelinquishFocus` method should return your part to the state where it possessed the focus before continuing to use that focus; how it does this is dependent on your part editor and the type of focus relinquished. Calling this method should have no effect if your part's `BeginRelinquishFocus` method returned `kODFalse`.

-----

## AbortRelinquishFocus - Exception Handling

`kODErrInvalidFrame`

The specified frame is not a display frame of this part.

-----

## AbortRelinquishFocus - Override Policy

If you subclass `ODPart`, you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

-----

## AbortRelinquishFocus - Related Methods

### Related Methods

- `ODPart::BeginRelinquishFocus`
- `ODPart::CommitRelinquishFocus`
- `ODSession::Tokenize`

-----

## AbortRelinquishFocus - Topics

### Class:

`ODPart`

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## AcquireContainingPartProperties

---

### AcquireContainingPartProperties - Syntax

This method writes into a storage unit the container properties this part associates with the specified embedded frame and then returns the storage unit.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODFrame          *frame;
ODStorageUnit    *rc;

rc = AcquireContainingPartProperties(frame);
```

---

### AcquireContainingPartProperties Parameter - frame

**frame** (ODFrame \*) - input  
A reference to an embedded frame of this part.

---

### AcquireContainingPartProperties Return Value - rc

**rc** (ODStorageUnit \*) - returns  
A reference to the storage unit that contains the container properties for the specified frame.

---

### AcquireContainingPartProperties - Parameters

**frame** (ODFrame \*) - input  
A reference to an embedded frame of this part.



**rc** (ODStorageUnit \*) - returns

A reference to the storage unit that contains the container properties for the specified frame.

-----

## AcquireContainingPartProperties - Remarks

The part embedded in this part calls this method to request the container properties that this part associates with the embedded part's display frame.

Before returning the storage unit object, your part's `AcquireContainingPartProperties` method should call the storage unit object's [Acquire](#) method. When the caller has finished using the returned storage unit, it should call the storage unit object's [Release](#) method.

-----

## AcquireContainingPartProperties - Exception Handling

`kODErrCannotEmbed`

This part does not support embedding.

`kODErrInvalidFrame`

The specified frame is not an embedded frame of this part.

-----

## AcquireContainingPartProperties - Override Policy

If you subclass [ODPart](#), you can override this method. Your override method must not call its inherited method; that is, your method must implement this method's functionality completely. This method needs to be implemented only by container parts.

-----

## AcquireContainingPartProperties - Related Methods

### Related Methods

- [ODPart::ContainingPartPropertiesUpdated](#)

-----

## AcquireContainingPartProperties - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

[Related Methods](#)

-----

# AdjustBorderShape

---

## AdjustBorderShape - Syntax

This method adjusts the shape of an embedded frame's active frame border.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODFacet      *embeddedFacet;
ODShape      *shape;
ODShape      *rc;

rc = AdjustBorderShape(embeddedFacet, shape);
```

---

## AdjustBorderShape Parameter - embeddedFacet

**embeddedFacet** (ODFacet \*) - input

A reference to an embedded facet whose active frame border is to be adjusted.

---

## AdjustBorderShape Parameter - shape

**shape** (ODShape \*) - input

A reference to a shape object defining the current active frame border or KODNULL if the active frame border is transferred to an embedded part of another part.

---

## AdjustBorderShape Return Value - rc

**rc** (ODShape \*) - returns

A reference to a revised shape object to use for the embedded frame's active frame border, clipped by this part's content and used shape.

---

## AdjustBorderShape - Parameters

**embeddedFacet** (ODFacet \*) - input

A reference to an embedded facet whose active frame border is to be adjusted.

**shape** (ODShape \*) - input

A reference to a shape object defining the current active frame border or kODNULL if the active frame border is transferred to an embedded part of another part.

**rc** (ODShape \*) - returns

A reference to a revised shape object to use for the embedded frame's active frame border, clipped by this part's content and used shape.

-----

## AdjustBorderShape - Remarks

OpenDoc calls this method when a frame embedded in this part has the selection focus. OpenDoc draws the active frame border, associated with the frame's facet, on the border of the facet's frame. Your part's AdjustBorderShape method may be called multiple times for a containing part that has several facets for the same embedded frame.

Your part's AdjustBorderShape method should clip the specified shape to account for content elements (including other embedded frames) that obscure portions of the active frame border, and then return the shape. Before returning the shape object, your part's AdjustBorderShape method should call the shape object's [Acquire](#) method (unless it modifies the requested shape object and returns that same object). When the caller has finished using the returned shape object, it should call the shape object's [Release](#) method.

Your part's AdjustBorderShape method should also adjust the clip shape of your part's facet to account for portions of the active frame border that obscure your content.

-----

## AdjustBorderShape - Exception Handling

kODErrCannotEmbed

This part does not support embedding.

kODErrInvalidFacet

The specified facet is not an embedded facet of this part.

-----

## AdjustBorderShape - Override Policy

If you subclass [ODPart](#), you can override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely. This method needs to be implemented only by container parts.

-----

## AdjustBorderShape - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

-----

# AdjustMenus

---

## AdjustMenus - Syntax

This method should prepare this part's menus for display.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODFrame      *frame;

AdjustMenus(frame);
```

---

## AdjustMenus Parameter - frame

**frame** (ODFrame \*) - input  
A reference to a display frame whose menus are to be displayed.

---

## AdjustMenus - Return Value

None.

---

## AdjustMenus - Parameters

**frame** (ODFrame \*) - input  
A reference to a display frame whose menus are to be displayed.

None.

---

## AdjustMenus - Remarks

OpenDoc calls this method when this part has the menu focus or is the root part and when there is a mouse-down event in the menu bar.

Your part's `AdjustMenus` method should perform any actions necessary to enable and disable menu items as appropriate to its current state. Your part's `AdjustMenus` method can use the [ODMenuBar](#) methods to change the appearance of your menu items, or it can make platform-specific calls to directly enable, disable, mark or change the text of your menu items.

---

## AdjustMenus - Exception Handling

`kODErrInvalidFrame`

The specified frame is not a display frame of this part.

---

## AdjustMenus - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## AdjustMenus - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

---

## AttachSourceFrame

---

## AttachSourceFrame - Syntax

This method should associate a source frame with a display frame for this part.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODFrame      *frame;
ODFrame      *sourceFrame;

AttachSourceFrame(frame, sourceFrame);
```

---

## AttachSourceFrame Parameter - frame

**frame** (ODFrame \*) - input  
A reference to a display frame for this part.

---

## AttachSourceFrame Parameter - sourceFrame

**sourceFrame** (ODFrame \*) - input  
A reference to a display frame to be used as the source for the specified display frame.

---

## AttachSourceFrame - Return Value

None.

---

## AttachSourceFrame - Parameters

**frame** (ODFrame \*) - input  
A reference to a display frame for this part.

**sourceFrame** (ODFrame \*) - input  
A reference to a display frame to be used as the source for the specified display frame.

None.

---

## AttachSourceFrame - Remarks

This part's containing part calls this method. This method tells this part to keep two or more of its display frames synchronized. The containing part calls this method immediately after creating a new display frame for this part.

Your part's AttachSourceFrame method should perform any actions necessary to synchronize the frames. If the two frames have identical presentations, your method should copy all embedded frames in the source frame into other frames (and attach the copied embedded frames to their source frames by calling the embedded part's AttachSourceFrame method).

---

## AttachSourceFrame - Exception Handling

kODErrInvalidFrame

The specified frame is not a display frame of this part.

---

## AttachSourceFrame - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## AttachSourceFrame - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

---

## BeginRelinquishFocus

---

## BeginRelinquishFocus - Syntax

This method should indicate whether this part is willing to relinquish the requested focus and should prepare this part to do so.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODTypeToken    focus;
ODFrame        *ownerFrame;
ODFrame        *proposedFrame;
ODBoolean      rc;

rc = BeginRelinquishFocus(focus, ownerFrame,
                          proposedFrame);
```

---

## BeginRelinquishFocus Parameter - focus

**focus** ([ODTypeToken](#)) - input

A tokenized string representing the focus type to be relinquished.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type.

You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the clipboard focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus

The frame with menu focus receives menu events.

kODModalFocus

A frame with modal focus is notifying other frames that it is the only currently modal frame.

kODMouseFocus

The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

kODScrollingFocus

The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

kODSelectionFocus

The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

-----

## BeginRelinquishFocus Parameter - ownerFrame

**ownerFrame** (ODFrame \*) - input

A reference to a display frame that currently possesses the focus.

-----

## BeginRelinquishFocus Parameter - proposedFrame

**proposedFrame** (ODFrame \*) - input

A reference to a frame requesting ownership of the focus.

-----

## BeginRelinquishFocus Return Value - rc

**rc** (ODBoolean) - returns

A flag indicating whether this part is willing to relinquish the requested focus.

kODTrue

This part is willing to relinquish the requested focus.

kODFalse

This part is not willing to relinquish the requested focus.

-----

## BeginRelinquishFocus - Parameters

**focus** (ODTypeToken) - input

A tokenized string representing the focus type to be relinquished.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.



kODClipboardFocus	The frame with the clipboard focus has access to the clipboard.
kODKeyFocus	The frame with keyboard focus receives keyboard events (excluding page-movement key events).
kODMenuFocus	The frame with menu focus receives menu events.
kODModalFocus	A frame with modal focus is notifying other frames that it is the only currently modal frame.
kODMouseFocus	The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.
kODScrollingFocus	The frame with the scrolling focus receives page-movement key events (such as <b>PageUp</b> ). This frame does not need to own keyboard focus.
kODSelectionFocus	The frame with the selection focus receives modified mouse-click events ( <b>Shift</b> +click and <b>Ctrl</b> +click). OpenDoc draws the active frame border around all facets of this frame.

**ownerFrame** (ODFrame \*) - input  
A reference to a display frame that currently possesses the focus.

**proposedFrame** (ODFrame \*) - input  
A reference to a frame requesting ownership of the focus.

**rc** (ODBoolean) - returns  
A flag indicating whether this part is willing to relinquish the requested focus.

kODTrue	This part is willing to relinquish the requested focus.
kODFalse	This part is not willing to relinquish the requested focus.

## BeginRelinquishFocus - Remarks

OpenDoc calls this method when ownership of the specified focus is requested and this part is the current owner. Upon receiving this call, this part decides whether it can relinquish the specified focus. Each part has part-specific requirements that must be met before it can relinquish a focus. If these requirements are met, this part should be ready to relinquish the focus.

Your part's `BeginRelinquishFocus` method should return either `kODTrue` or `kODFalse`, based on the type of focus and the identities of the frames (current and proposed focus owners) passed in. In most cases, your method should return `kODTrue`; however, your method might return `kODFalse`, for example, if your part is displaying a modal dialog box and another part is requesting the modal focus. In that case, because your part does not want to yield the modal focus until its modal dialog box closes, your method should return `kODFalse`. In situations when your method does return `kODFalse`, your part's `AbortRelinquishFocus` method does not need to take any action.

Your part's `BeginRelinquishFocus` method should also ensure that, during the process of relinquishing the specified focus and before your part's `CommitRelinquishFocus` method is called, your part does not enter a state where the focus cannot be relinquished.

## BeginRelinquishFocus - Exception Handling

kODErrInvalidFrame	The specified owner frame is not a display frame of this part.
--------------------	--

## BeginRelinquishFocus - Override Policy

If you subclass `ODPart`, you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

# BeginRelinquishFocus - Related Methods

## Related Methods

- [ODPart::AbortRelinquishFocus](#)
  - [ODPart::CommitRelinquishFocus](#)
  - [ODSession::Tokenize](#)
- 

# BeginRelinquishFocus - Topics

## Class:

[ODPart](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

[Related Methods](#)

---

# BreakLink (OS/2)

---

## BreakLink (OS/2) - Syntax

This method should indicate that the user has chosen, in the *Link* page of the Properties notebook, to break a link connection.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODLink      *link;

BreakLink(link);
```

---

## BreakLink (OS/2) Parameter - link

**link** (ODLink \*) - input

The link object to be disconnected from its link source.

---

## BreakLink (OS/2) - Return Value

None.

---

## BreakLink (OS/2) - Parameters

**link** (ODLink \*) - input

The link object to be disconnected from its link source.

None.

---

## BreakLink (OS/2) - Remarks

There is no default action in this method except to raise an exception, `kODErrSubClassResponsibility`.

Your part should unregister and release the link object and update any tables maintained by the part which associated the link object with the link content. This includes identifying embedded frames within this content as no longer being part of this link target and calling the [ChangeLinkStatus](#) method.

---

## BreakLink (OS/2) - Exception Handling

`kODErrSubClassResponsibility`

The subclass must override this method.

---

## BreakLink (OS/2) - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must call the link's [Release](#) method.

---

## BreakLink (OS/2) - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

## BreakLinkSource (OS/2)

---

## BreakLinkSource (OS/2) - Syntax

This method should indicate that the user has chosen, in the *Link Source* page of the Properties notebook, to break a link source from all connected links.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODLinkSource      *linkSource;

BreakLinkSource(linkSource);
```

---

## BreakLinkSource (OS/2) Parameter - linkSource

**linkSource** (ODLinkSource \*) - input  
The link source object to be discarded.

---

## BreakLinkSource (OS/2) - Return Value

None.

---

## BreakLinkSource (OS/2) - Parameters

**linkSource** (ODLinkSource \*) - input  
The link source object to be discarded.

None.

---

## BreakLinkSource (OS/2) - Remarks

There is no default action in this method except to raise an exception `KODerrSubClassResponsibility`.

Your part should update any tables maintained by the part which associate the link-source object with the link-source content. This includes identifying embedded frames within this content as no longer being part of this link source and calling the [ChangeLinkStatus](#) method.

-----

## BreakLinkSource (OS/2) - Exception Handling

`KODerrSubClassResponsibility`

The subclass must override this method.

-----

## BreakLinkSource (OS/2) - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must call the link source's [SetSourcePart](#) method with the *sourcePartSU* parameter set to `KODNULL` and then release the link-source object.

-----

## BreakLinkSource (OS/2) - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

-----

## CanvasChanged

-----

## CanvasChanged - Syntax

This method should transfer asynchronous imaging to a new canvas.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>
```

```
ODFacet      *facet;
```

```
CanvasChanged(facet);
```

---

## CanvasChanged Parameter - facet

**facet** (ODFacet \*) - input  
A reference to a facet with the new canvas.

---

## CanvasChanged - Return Value

None.

---

## CanvasChanged - Parameters

**facet** (ODFacet \*) - input  
A reference to a facet with the new canvas.

None.

---

## CanvasChanged - Remarks

OpenDoc calls this method if an offscreen canvas is assigned to, changed, or removed from your part's facet while your part is running. Your part must prepare to draw on the new canvas, update the cached information, and alter asynchronous display information.

---

## CanvasChanged - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## CanvasChanged - Topics

**Class:**  
ODPart

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)

---

# CanvasUpdated

---

## CanvasUpdated - Syntax

This method should copy the content of the specified canvas (owned by this part) to its parent canvas.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODCanvas      *canvas;

CanvasUpdated(canvas);
```

---

## CanvasUpdated Parameter - canvas

**canvas** (ODCanvas \*) - input  
A reference to a canvas that is to be updated.

---

## CanvasUpdated - Return Value

None.

---

## CanvasUpdated - Parameters

**canvas** (ODCanvas \*) - input  
A reference to a canvas that is to be updated.

None.

---

## CanvasUpdated - Remarks

Your part's `CanvasUpdated` method is called when an area of a canvas owned by this part changes and needs to be updated. The owning part's facet calls this method to allow the owning part to copy the image of the offscreen canvas to its parent canvas during updates.

---

## CanvasUpdated - Exception Handling

`kODErrCannotEmbed`

This part does not support embedding.

---

## CanvasUpdated - Override Policy

If you subclass [ODPart](#) you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely. This method needs to be implemented only by container parts.

---

## CanvasUpdated - Topics

### Class:

`ODPart`

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

---

## ChangeKind

---

## ChangeKind - Syntax

This method should change this part's preferred kind.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>
```

```
ODType    kind;
```

```
ChangeKind(kind);
```

---



## ChangeKind Parameter - kind

**kind** (ODType) - input

The preferred kind to be assigned to this part.

---

## ChangeKind - Return Value

None.

---

## ChangeKind - Parameters

**kind** (ODType) - input

The preferred kind to be assigned to this part.

---

None.

---

## ChangeKind - Remarks

OpenDoc calls this method if the user has chosen a new preferred kind for this part in the Properties notebook (for example, a request that a Styled Text part change into an ASCII part).

After your part's ChangeKind method executes successfully, your part should begin using the specified part kind as its preferred kind and begin manipulating your part's data in the new format. Your part should write the preferred kind into a kODPropPreferredKind property in its storage unit and store its part kinds in order from highest fidelity to lowest.

---

## ChangeKind - Exception Handling

kODErrUnsupportedKind

The part does not support that kind of content. (This exception is supported by the part developer.)

---

## ChangeKind - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely. Your override method should always be implemented, not just implemented in the case of a part supporting multiple kinds.

---

# ChangeKind - Topics

## Class:

ODPart

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

---

## ClonePartInfo

---

## ClonePartInfo - Syntax

This method should clone a display frame's part-information data into the specified storage-unit view object.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODDraftKey      key;
ODInfoType      partInfo;
ODStorageUnitView *storageUnitView;
ODFrame         *scope;

ClonePartInfo(key, partInfo, storageUnitView,
              scope);
```

---

## ClonePartInfo Parameter - key

**key** ([ODDraftKey](#)) - input

The draft key identifying a particular cloning operation. The key provides thread-safe access to cloning.

---

## ClonePartInfo Parameter - partInfo

**partInfo** ([ODInfoType](#)) - input

The part-information data to be cloned.

---

# ClonePartInfo Parameter - storageUnitView

**storageUnitView** (ODStorageUnitView \*) - input

A reference to a storage-unit view object that is focused to the frame's part-information property, but not to any value within that property.

---

# ClonePartInfo Parameter - scope

**scope** (ODFrame \*) - input

A reference to a display frame that defines the scope of the cloning operation. All cloned storage units must be within the scope of the specified frame.

---

# ClonePartInfo - Return Value

None.

---

# ClonePartInfo - Parameters

**key** (ODDraftKey) - input

The draft key identifying a particular cloning operation. The key provides thread-safe access to cloning.

**partInfo** (ODInfoType) - input

The part-information data to be cloned.

**storageUnitView** (ODStorageUnitView \*) - input

A reference to a storage-unit view object that is focused to the frame's part-information property, but not to any value within that property.

**scope** (ODFrame \*) - input

A reference to a display frame that defines the scope of the cloning operation. All cloned storage units must be within the scope of the specified frame.

None.

---

# ClonePartInfo - Remarks

When a document is saved as a copy (the **Save a Copy** command from the **Document** menu) or data is transferred (using drag-and-drop or linking), OpenDoc may ask your part to save a display frame's part-information data to a particular storage unit. OpenDoc calls this method when it writes the part-information data for a display frame of your part.

Your part's ClonePartInfo method must get the storage unit associated with the specified storage-unit view and focus it to the values in the

frame's part-information property as necessary to write the formats it needs.

Your part's ClonePartInfo method should write out the part-information data. Your part's ClonePartInfo method should also clone any additional objects to which your part has strong persistent references and that are within the specified display frame's scope.

-----

## ClonePartInfo - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

-----

## ClonePartInfo - Related Methods

### Related Methods

- [ODPart::ReadPartInfo](#)
- [ODPart::WritePartInfo](#)

-----

## ClonePartInfo - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Related Methods](#)

-----

## CommitRelinquishFocus

-----

## CommitRelinquishFocus - Syntax

This method should complete this part's relinquishing of ownership of the specified focus.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODTypeToken    focus;
ODFrame        *ownerFrame;
ODFrame        *proposedFrame;

CommitRelinquishFocus(focus, ownerFrame, proposedFrame);
```

---

## CommitRelinquishFocus Parameter - focus

**focus** (ODTypeToken) - input

A tokenized string representing the focus type to be relinquished.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the clipboard focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus

The frame with menu focus receives menu events.

kODModalFocus

A frame with modal focus is notifying other frames that it is the only currently modal frame.

kODMouseFocus

The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

kODScrollingFocus

The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

kODSelectionFocus

The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

---

## CommitRelinquishFocus Parameter - ownerFrame

**ownerFrame** (ODFrame \*) - input

A reference to a display frame that currently possesses the focus.

---

## CommitRelinquishFocus Parameter - proposedFrame

**proposedFrame** (ODFrame \*) - input

A reference to a frame that requested the focus.

---

## CommitRelinquishFocus - Return Value

None.

---

## CommitRelinquishFocus - Parameters

**focus** ([ODTypeToken](#)) - input

A tokenized string representing the focus type to be relinquished.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

**kODClipboardFocus**

The frame with the clipboard focus has access to the clipboard.

**kODKeyFocus**

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

**kODMenuFocus**

The frame with menu focus receives menu events.

**kODModalFocus**

A frame with modal focus is notifying other frames that it is the only currently modal frame.

**kODMouseFocus**

The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

**kODScrollingFocus**

The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

**kODSelectionFocus**

The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

**ownerFrame** ([ODFrame \\*](#)) - input

A reference to a display frame that currently possesses the focus.

**proposedFrame** ([ODFrame \\*](#)) - input

A reference to a frame that requested the focus.

None.

-----

## CommitRelinquishFocus - Remarks

Your part's `CommitRelinquishFocus` method should complete the process initiated when OpenDoc first called your part's [BeginRelinquishFocus](#) method. If all focus owners in a given request for a focus set are willing to relinquish focus, OpenDoc calls each part's `CommitRelinquishFocus` method.

Your part's `CommitRelinquishFocus` method should perform any actions necessary to relinquish the specified focus to OpenDoc. Some actions depend on the nature and implementation of the part itself, but others are standard. Your part's `CommitRelinquishFocus` method should remove menus or palettes of your part's frame, remove highlighting, and relinquish any external resources associated with the specified focus. Remember that the focus might be moving from one frame to another of the same part, so the exact actions can vary.

-----

## CommitRelinquishFocus - Exception Handling

**kODErrInvalidFrame**

The specified frame is not a display frame of this part.

-----

## CommitRelinquishFocus - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

-----

# CommitRelinquishFocus - Related Methods

## Related Methods

- [ODPart::AbortRelinquishFocus](#)
- [ODPart::BeginRelinquishFocus](#)
- [ODSession::Tokenize](#)

---

# CommitRelinquishFocus - Topics

## Class:

[ODPart](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Exception Handling](#)  
[Related Methods](#)

---

# ContainingPartPropertiesUpdated

---

# ContainingPartPropertiesUpdated - Syntax

This method should notify this part that its containing part's container properties have changed.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODFrame          *frame;
ODStorageUnit    *propertyUnit;

ContainingPartPropertiesUpdated(frame, propertyUnit);
```

---

# ContainingPartPropertiesUpdated Parameter - frame

**frame** (ODFrame \*) - input

A reference to a display frame to which the container properties apply.

# ContainingPartPropertiesUpdated Parameter - propertyUnit

**propertyUnit** (ODStorageUnit \*) - input

A reference to a storage unit that contains the changed container properties.

---

## ContainingPartPropertiesUpdated - Return Value

None.

---

## ContainingPartPropertiesUpdated - Parameters

**frame** (ODFrame \*) - input

A reference to a display frame to which the container properties apply.

**propertyUnit** (ODStorageUnit \*) - input

A reference to a storage unit that contains the changed container properties.

None.

---

## ContainingPartPropertiesUpdated - Remarks

This part's containing part calls this method when the container properties it provides for adoption have changed. Your part's ContainingPartPropertiesUpdated method should inspect the specified storage unit for container properties that this part can understand. Where applicable, your method should adopt those container properties for its own appearance or behavior. It ignores inapplicable container properties, without generating an exception.

---

## ContainingPartPropertiesUpdated - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## ContainingPartPropertiesUpdated - Related Methods

### Related Methods

- [ODPart::AcquireContainingPartProperties](#)



---

# ContainingPartPropertiesUpdated - Topics

## Class:

ODPart

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Related Methods](#)

---

## CreateEmbeddedFramesIterator

---

### CreateEmbeddedFramesIterator - Syntax

This method creates an embedded-frames iterator to give callers access to all embedded frames of this part.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODFrame          *frame;
ODEmbeddedFramesIterator *rc;

rc = CreateEmbeddedFramesIterator(frame);
```

---

### CreateEmbeddedFramesIterator Parameter - frame

**frame** (ODFrame \*) - input

A reference to a display frame whose embedded frames the iterator traverses.

---

### CreateEmbeddedFramesIterator Return Value - rc

**rc** (ODEmbeddedFramesIterator \*) - returns

A reference to a new embedded-frames iterator object.

---

### CreateEmbeddedFramesIterator - Parameters

**frame** (ODFrame \*) - input  
A reference to a display frame whose embedded frames the iterator traverses.

**rc** (ODEmbeddedFramesIterator \*) - returns  
A reference to a new embedded-frames iterator object.

---

## CreateEmbeddedFramesIterator - Remarks

Your part's `CreateEmbeddedFramesIterator` method should create, initialize, and return an embedded-frames iterator. Your part calls this method if it needs to apply an operation to all frames directly embedded within a display frame of another part. It is your responsibility to delete the iterator when it is no longer needed.

While you are using an embedded-frames iterator, you should not modify the list of embedded frames for the part. You must postpone adding items to or removing items from the list of embedded frames for the part until after you have deleted the iterator.

---

## CreateEmbeddedFramesIterator - Exception Handling

<code>kODErrCannotEmbed</code>	This part does not support embedding.
<code>kODErrInvalidFrame</code>	This specified frame is not a display frame of this part.
<code>kODErrOutOfMemory</code>	There is not enough memory to allocate embedded-frames iterator object.

---

## CreateEmbeddedFramesIterator - Override Policy

If you subclass [ODPart](#), you can override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely. This method needs to be implemented only by container parts.

---

## CreateEmbeddedFramesIterator - Topics

**Class:**  
[ODPart](#)

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Exception Handling](#)

---

## CreateLink

---

# CreateLink - Syntax

This method creates a link-source object for this part.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODByteArray      *data;
ODLinkSource     *rc;

rc = CreateLink(data);
```

---

## CreateLink Parameter - data

**data** (ODByteArray \*) - input

A byte array whose buffer contains the data of a link specification, created earlier by this part, that defines the content of the link source to be linked to.

---

## CreateLink Return Value - rc

**rc** (ODLinkSource \*) - returns

A reference to a new link-source object to be used to represent the data.

---

## CreateLink - Parameters

**data** (ODByteArray \*) - input

A byte array whose buffer contains the data of a link specification, created earlier by this part, that defines the content of the link source to be linked to.

**rc** (ODLinkSource \*) - returns

A reference to a new link-source object to be used to represent the data.

---

## CreateLink - Remarks

OpenDoc calls this method when the user decides to create a link when pasting in or dropping data that originated in this part. If a link already exists to the specified source content, this method should return the existing link-source object; otherwise, this method should create a new link-source object and copy the source content into it.

The data in a link specification is private to the part writing it. The data is only returned to this part when the CreateLink method actually creates a link-source object. If your part creates a link-source object, your part must maintain information about what portion of its content has been linked so that it may update the link-source object when the content data changes.

Before returning the link-source object, your part's `CreateLink` method should call the link-source object's [Acquire](#) method. When the caller has finished using the returned link-source object, it should call the link-source object's [Release](#) method.

---

## CreateLink - Exception Handling

`kODErrDoesNotLink`  
`kODErrOutOfMemory`

This part does not support linking.  
There is not enough memory to allocate the link-source object.

---

## CreateLink - Override Policy

If you subclass [ODPart](#), you can override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## CreateLink - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

---

## CreateRootMenuBar (OS/2)

---

## CreateRootMenuBar (OS/2) - Syntax

This method creates a menu bar for a root frame to be used as a basis for the active frame's menu bar.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>
```

```
ODFrame      *frame;
ODMenuBar    *menuBar;
```

```
menuBar = CreateRootMenuBar(frame);
```

---

# CreateRootMenuBar (OS/2) Parameter - frame

**frame** (ODFrame \*) - input

A pointer to the root frame of the window for which the menu bar is to be created.

---

# CreateRootMenuBar (OS/2) Return Value - menuBar

**menuBar** (ODMenuBar \*) - returns

The menu bar to be used as the base menu bar for this window.

The value returned from the window-state object's [CopyBaseMenuBar](#) method should become the window's base menu bar.

The caller of this method is responsible for releasing the menu bar when it has finished using it. If this method passes out several references to the same menu bar, this method increments the reference count of that menu-bar object for each reference after the first.

---

# CreateRootMenuBar (OS/2) - Parameters

**frame** (ODFrame \*) - input

A pointer to the root frame of the window for which the menu bar is to be created.

**menuBar** (ODMenuBar \*) - returns

The menu bar to be used as the base menu bar for this window.

The value returned from the window-state object's [CopyBaseMenuBar](#) method should become the window's base menu bar.

The caller of this method is responsible for releasing the menu bar when it has finished using it. If this method passes out several references to the same menu bar, this method increments the reference count of that menu-bar object for each reference after the first.

---

# CreateRootMenuBar (OS/2) - Exception Handling

kODErrOutOfMemory

There is not enough memory to allocate the menu bar.

---

# CreateRootMenuBar (OS/2) - Topics

**Class:**

ODPart

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Exception Handling](#)

---

# DisplayFrameAdded

---

## DisplayFrameAdded - Syntax

This method should add the specified frame to this part's internal list of display frames.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODFrame      *frame;

DisplayFrameAdded(frame);
```

---

## DisplayFrameAdded Parameter - frame

**frame** (ODFrame \*) - input  
A reference to a display frame to be added.

---

## DisplayFrameAdded - Return Value

None.

---

## DisplayFrameAdded - Parameters

**frame** (ODFrame \*) - input  
A reference to a display frame to be added.

None.

---

## DisplayFrameAdded - Remarks

OpenDoc calls this method only when a new display frame has been added to this part during frame object initialization.

Your part's `DisplayFrameAdded` method should perform any actions necessary to handle the addition of the specified frame to your part's internal list of display frames. Some actions depend on the nature and implementation of the part itself, but others are standard, such as adding the new frame to your part's internal list of display frames, verifying that you can support the frame's view type and presentation, and adding any needed part-information data to the frame.

Your part should not perform any layout or imaging tasks as a result of a display frame being added; it should wait until your part's `FacetAdded` method is called, at which time the frame becomes visible.

Your part's `DisplayFrameAdded` method is typically called after `OpenDoc` or this part's containing part calls a draft's `CreateFrame` method. By contrast, a part's `DisplayFrameConnected` method is called after `OpenDoc` or this part's containing part calls a draft's `AcquireFrame` method.

---

## DisplayFrameAdded - Override Policy

If you subclass `ODPart`, you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## DisplayFrameAdded - Related Methods

### Related Methods

- [ODPart::DisplayFrameConnected](#)

---

## DisplayFrameAdded - Topics

### Class:

`ODPart`

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Related Methods](#)

---

## DisplayFrameClosed

---

## DisplayFrameClosed - Syntax

This method should notify this part that one of its display frames has been closed.

```
#define INCL_ODPART
#define INCL_ODAPI
```

```
#include <os2.h>

ODFrame      *frame;

DisplayFrameClosed(frame);
```

---

## DisplayFrameClosed Parameter - frame

**frame** (ODFrame \*) - input  
A reference to a display frame to be closed.

---

## DisplayFrameClosed - Return Value

None.

---

## DisplayFrameClosed - Parameters

**frame** (ODFrame \*) - input  
A reference to a display frame to be closed.

None.

---

## DisplayFrameClosed - Remarks

OpenDoc calls this method when this part's document or window closes.

Your part's DisplayFrameClosed method should update your part's internal list of display frames and other related structures to reflect the removal of the specified display frame from memory, and it should close any embedded frames. In addition, your method should call the arbitrator's [RelinquishFocusSet](#) method to relinquish any foci owned by the display frame being closed.

Your part's DisplayFrameClosed method should not alter a part's stored list of displayed frames; it reflects only the deletion of the currently instantiated frame object. By contrast, OpenDoc calls a part's [DisplayFrameRemoved](#) method to permanently remove one of the part's display frames.

---

## DisplayFrameClosed - Exception Handling

kODErrInvalidFrame

The specified frame is not a display frame of this part.



---

## DisplayFrameClosed - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## DisplayFrameClosed - Related Methods

### Related Methods

- [ODPart::DisplayFrameRemoved](#)
- 

## DisplayFrameClosed - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## DisplayFrameConnected

---

## DisplayFrameConnected - Syntax

This method should add the specified frame to this part's internal list of display frames.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODFrame      *frame;

DisplayFrameConnected(frame);
```

---

## DisplayFrameConnected Parameter - frame

**frame** (ODFrame \*) - input  
A reference to a display frame to be connected.

---

## DisplayFrameConnected - Return Value

None.

---

## DisplayFrameConnected - Parameters

**frame** (ODFrame \*) - input  
A reference to a display frame to be connected.

None.

---

## DisplayFrameConnected - Remarks

OpenDoc calls this method when a previously stored display frame is initialized and reconnected to this part. When this part's document opens, or as each display frame of this part becomes visible through scrolling or resizing, OpenDoc calls this method when it or this part's containing part instantiates and connects one of this part's previously stored display frames.

Your part's DisplayFrameConnected method should update your part's internal list of display frames and other related structures to reflect the connection of the specified display frame. If the specified frame is already in the list, no action is taken. Your part should also update relevant information in the display frame, for example, its used shape and presentation.

Your part's DisplayFrameConnected method is typically called after OpenDoc or this part's containing part calls a draft's [AcquireFrame](#) method. By contrast, a part's [DisplayFrameAdded](#) method is called after OpenDoc or this part's containing part calls a draft's [CreateFrame](#) method.

---

## DisplayFrameConnected - Exception Handling

kODErrInvalidFrame

The specified frame is not a display frame of this part.

---

## DisplayFrameConnected - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

# DisplayFrameConnected - Related Methods

## Related Methods

- [ODPart::DisplayFrameAdded](#)

---

## DisplayFrameConnected - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

[Related Methods](#)

---

## DisplayFrameRemoved

---

## DisplayFrameRemoved - Syntax

This method should remove the specified frame from this part's internal list of display frames.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODFrame      *frame;

DisplayFrameRemoved(frame);
```

---

## DisplayFrameRemoved Parameter - frame

**frame** (ODFrame \*) - input

A reference to a display frame to be removed.

---

## DisplayFrameRemoved - Return Value

None.

---

## DisplayFrameRemoved - Parameters

**frame** (ODFrame \*) - input

A reference to a display frame to be removed.

None.

---

## DisplayFrameRemoved - Remarks

To permanently delete one of its embedded frames, this part's containing part calls the embedded frame's [Remove](#) method. After removing the embedded frame, OpenDoc calls this method to notify this part of the removal.

Your part's DisplayFrameRemoved method should perform any actions necessary to remove a frame. For example, it should remove any frames (and part information) embedded within the specified frame. Your part's DisplayFrameRemoved method should update your part's internal list of display frames and other related structures to reflect the removal of the specified display frame. In addition, your method should call the arbitrator's [RelinquishFocusSet](#) method to relinquish any foci owned by the display frame being removed.

By contrast, OpenDoc calls a part's [DisplayFrameClosed](#) method to clean up when the part's document or window closes.

---

## DisplayFrameRemoved - Exception Handling

kODErrInvalidFrame

The specified frame is not a display frame of this part.

---

## DisplayFrameRemoved - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## DisplayFrameRemoved - Related Methods

### Related Methods

- [ODPart::DisplayFrameClosed](#)
-

# DisplayFrameRemoved - Topics

## Class:

ODPart

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

[Related Methods](#)

---

## DisposeActionState

---

## DisposeActionState - Syntax

This method dispose of the undo-action data.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODActionData    *actionState;
ODDoneState     doneState;

DisposeActionState(actionState, doneState);
```

---

## DisposeActionState Parameter - actionState

**actionState** ([ODActionData](#) \*) - input

A byte array whose buffer contains the data previously logged by this part to allow it to undo the action.

---

## DisposeActionState Parameter - doneState

**doneState** ([ODDoneState](#)) - input

The state of the undo action. This parameter can be set to one of the following values:

kODDone	The actions was done and is now on the document's undo stack.
kODReDone	The action was done, undone, and redone and is now on the document's undo stack.
kODUndone	

The action was done and undone and is now on the document's redo stack.

---

## DisposeActionState - Return Value

None.

---

## DisposeActionState - Parameters

**actionState** ([ODActionData](#) \*) - input

A byte array whose buffer contains the data previously logged by this part to allow it to undo the action.

**doneState** ([ODDoneState](#)) - input

The state of the undo action. This parameter can be set to one of the following values:

kODDone	The actions was done and is now on the document's undo stack.
kODReDone	The action was done, undone, and redone and is now on the document's undo stack.
kODUndone	The action was done and undone and is now on the document's redo stack.

None.

---

## DisposeActionState - Remarks

OpenDoc calls this method when it disposes of the undo-action data. After this method executes successfully, memory for the action state is reclaimed and is no longer usable for undo operations, effectively committing the action.

---

## DisposeActionState - Exception Handling

kODErrDoesNotUndo

The specified action is not an undo action of this part.

---

## DisposeActionState - Override Policy

If you subclass [ODPart](#), you can override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## DisposeActionState - Related Methods

## Related Methods

- [ODPart::ReadActionState](#)
- [ODPart::WriteActionState](#)

---

# DisposeActionState - Topics

## Class:

ODPart

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

[Related Methods](#)

---

# DragEnter

---

## DragEnter - Syntax

This method begins tracking a drag operation.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODDragItemIterator    *dragInfo;
ODFacet               *facet;
ODPoint               *where;
ODDragResult          rc;

rc = DragEnter(dragInfo, facet, where);
```

---

## DragEnter Parameter - dragInfo

**dragInfo** (ODDragItemIterator \*) - input

A reference to a drag-item iterator that describes the content, as well as the type and values, of the dragged data.

---

## DragEnter Parameter - facet

**facet** (ODFacet \*) - input  
A reference to a facet where the drag point is located.

---

## DragEnter Parameter - where

**where** (ODPoint \*) - input  
The location of the drag point, expressed in frame coordinates.

---

## DragEnter Return Value - rc

**rc** (ODDragResult) - returns  
A flag indicating whether this part can accept the dragged data in the given facet.

kODTrue	This part can accept the dragged data in the given facet.
kODFalse	This part cannot accept the dragged data in the given facet.

---

## DragEnter - Parameters

**dragInfo** (ODDragItemIterator \*) - input  
A reference to a drag-item iterator that describes the content, as well as the type and values, of the dragged data.

**facet** (ODFacet \*) - input  
A reference to a facet where the drag point is located.

**where** (ODPoint \*) - input  
The location of the drag point, expressed in frame coordinates.

**rc** (ODDragResult) - returns  
A flag indicating whether this part can accept the dragged data in the given facet.

kODTrue	This part can accept the dragged data in the given facet.
kODFalse	This part cannot accept the dragged data in the given facet.

---

## DragEnter - Remarks

OpenDoc calls this method when the drag point has entered the specified facet of this part during a drag operation. Before calling this method, OpenDoc calls the [IsDraggable](#) method of the facet's frame to ensure that this part is able to receive a drop.

Your part's DragEnter method should examine the part kinds of the dragged data (using the drag-item iterator specified by the *dragInfo* parameter), and determine whether it can accept the dragged data. Your part should not at this point attempt to read data from any of the storage units supplied by the drag-item iterator. If your part can accept a drop, it should provide the appropriate feedback to the user, such as



displaying its drag border or changing the cursor appearance. If your part cannot handle the part kinds of the dragged data, it should take no action. If your part's `DragEnter` method returns `kODFalse`, OpenDoc assumes your part cannot accept a drop. OpenDoc will not make additional calls to your [DragWithin](#) and [Drop](#) methods as long as the mouse pointer remains in this facet.

---

## DragEnter - Exception Handling

`kODErrDoesNotDrop`

This part does not support drag and drop.

---

## DragEnter - Override Policy

If you subclass [ODPart](#), you can override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## DragEnter - Related Methods

### Related Methods

- [ODPart::DragLeave](#)
  - [ODPart::DragWithin](#)
- 

## DragEnter - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

[Related Methods](#)

---

## DragLeave

---

## DragLeave - Syntax

This method finishes tracking a drag operation and deactivates this part from the drag tracking.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODFacet      *facet;
ODPoint      *where;

DragLeave(facet, where);
```

---

## DragLeave Parameter - facet

**facet** (ODFacet \*) - input  
A reference to a facet where the drag point is located.

---

## DragLeave Parameter - where

**where** (ODPoint \*) - input  
The location of the drag point, expressed in frame coordinates.

---

## DragLeave - Return Value

None.

---

## DragLeave - Parameters

**facet** (ODFacet \*) - input  
A reference to a facet where the drag point is located.

**where** (ODPoint \*) - input  
The location of the drag point, expressed in frame coordinates.

None.

---

## DragLeave - Remarks

OpenDoc calls this method when the drag point has left the specified facet of this part during a drag operation. After this method executes successfully, the part is guaranteed not to receive [DragWithin](#) or [Drop](#) messages until you receive another [DragEnter](#) message.

Your part's `DragLeave` method should clean up after a drag operation. For example, you should remove the drag border on the frame, remove highlighting from any content highlighted during drag tracking, and change the cursor appearance back to its original form.

---

## DragLeave - Exception Handling

`kODErrDoesNotDrop`

This part does not support drag and drop.

---

## DragLeave - Override Policy

If you subclass [ODPart](#), you can override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## DragLeave - Related Methods

### Related Methods

- [ODPart::DragEnter](#)
  - [ODPart::DragWithin](#)
- 

## DragLeave - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

[Related Methods](#)

---

## DragWithin

---

## DragWithin - Syntax

This method tracks a drag operation and provides graphical feedback regarding possible drop targets.

```

#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODDragItemIterator    *dragInfo;
ODFacet               *facet;
ODPoint               *where;
ODDragResult          rc;

rc = DragWithin(dragInfo, facet, where);

```

---

## DragWithin Parameter - dragInfo

**dragInfo** (ODDragItemIterator \*) - input

A reference to a drag-item iterator that describes the content, as well as the types and values, of the dragged data.

---

## DragWithin Parameter - facet

**facet** (ODFacet \*) - input

A reference to a facet where the drag point is located.

---

## DragWithin Parameter - where

**where** (ODPoint \*) - input

The location of the drag point, expressed in frame coordinates.

---

## DragWithin Return Value - rc

**rc** (ODDragResult) - returns

A flag indicating whether this part can accept the dragged data in the given facet.

kODTrue

This part can accept the dragged data in the given facet.

kODFalse

This part cannot accept the dragged data in the given facet.

---

## DragWithin - Parameters

**dragInfo** (ODDragItemIterator \*) - input

A reference to a drag-item iterator that describes the content, as well as the types and values, of the dragged data.

**facet** (ODFacet \*) - input

A reference to a facet where the drag point is located.

**where** (ODPoint \*) - input

The location of the drag point, expressed in frame coordinates.

**rc** (ODDragResult) - returns

A flag indicating whether this part can accept the dragged data in the given facet.

kODTrue

This part can accept the dragged data in the given facet.

kODFalse

This part cannot accept the dragged data in the given facet.

-----

## DragWithin - Remarks

Your part's DragWithin method allows your part to do any desired processing inside of its display frame. For example, if your part allows objects to be dropped only on individual hot spots, it may change its feedback based on mouse-pointer location. If the mouse is not over these hot spots, the cursor might need to be changed to reflect that no dropping can be done, even though the mouse is in a droppable frame. OpenDoc calls this method continuously when the drag point is within the borders of the specified facet's frame.

Your part's DragWithin method should examine the part kinds of the dragged data (using the drag-item iterator specified by the *dragInfo* parameter) and determine whether it can accept the dragged data. At this point, your part should not attempt to read data from any of the storage units supplied by the drag-item iterator. If your part can accept a drop, it should provide the appropriate feedback to the user, for example, by displaying its drag border or changing the cursor appearance. If your part cannot handle the part kinds of the dragged data, it should take no action. If your part's DragWithin method returns kODFalse, OpenDoc assumes you no longer wish to accept a drop in this facet. It will not make additional calls to your DragWithin or [Drop](#) methods as long as the mouse pointer remains in this facet.

Your part's DragWithin method also represents an opportunity for your part to examine the state of the machine. For example, some parts may want to know whether the modifier keys are down while a drag operation is in progress.

-----

## DragWithin - Exception Handling

kODErrDoesNotDrop

This part does not support drag and drop.

-----

## DragWithin - Override Policy

If you subclass [ODPart](#), you can override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

-----

## DragWithin - Related Methods

### Related Methods

- [ODPart::DragEnter](#)
- [ODPart::DragLeave](#)

---

# DragWithin - Topics

## Class:

ODPart

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

[Related Methods](#)

---

## Draw

---

## Draw - Syntax

This method should draw this part within the area that needs updating in the specified facet.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODFacet      *facet;
ODShape      *invalidShape;

Draw(facet, invalidShape);
```

---

## Draw Parameter - facet

**facet** (ODFacet \*) - input

A reference to the facet in which this part is to draw.

---

## Draw Parameter - invalidShape

**invalidShape** (ODShape \*) - input

A reference to a shape object defining the area of the facet that needs updating, expressed in frame coordinates.

---

# Draw - Return Value

None.

---

# Draw - Parameters

**facet** (ODFacet \*) - input

A reference to the facet in which this part is to draw.

**invalidShape** (ODShape \*) - input

A reference to a shape object defining the area of the facet that needs updating, expressed in frame coordinates.

None.

---

# Draw - Remarks

OpenDoc calls this method when an update event occurs that involves a facet of this part. Your part's Draw method should make the actual platform-specific drawing calls.

Your part's Draw method should draw this part's content on the facet's canvas, updating the portion of the facet specified in the invalid shape. Your part must determine whether it should draw on the screen or to a printer, and then draw itself appropriately.

---

# Draw - Exception Handling

kODErrInvalidFrame

The specified frame is not a display frames of this part.

---

# Draw - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

# Draw - Related Methods

## Related Methods

- [ODFacet::Draw](#)
- [ODFacet::Update](#)
- [ODWindow::Update](#)

---

# Draw - Topics

## Class:

ODPart

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

[Related Methods](#)

---

## Drop

---

## Drop - Syntax

This method moves or copies the dragged data in to this part.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODDragItemIterator    *dropInfo;
ODFacet               *facet;
ODPoint               *where;
ODDropResult          rc;

rc = Drop(dropInfo, facet, where);
```

---

## Drop Parameter - dropInfo

**dropInfo** (ODDragItemIterator \*) - input

A reference to a drag-item iterator that describes the content, as well as the types and values, of the dragged data.

---

## Drop Parameter - facet

**facet** (ODFacet \*) - input

A reference to a facet where the drop has occurred.



---

## Drop Parameter - where

**where** (ODPoint \*) - input

The location of the drop point, expressed in frame coordinates.

---

## Drop Return Value - rc

**rc** (ODDropResult) - returns

The result of the drop operation. This parameter can return one of the following values:

kODDropCopy

A synchronous copy operation was successful.

kODDropFail

A synchronous drop operation was unsuccessful.

kODDropMove

A synchronous move operation was successful

kODDropUnfinished

An asynchronous drop is in progress (on platforms that support asynchronous drops.)

---

## Drop - Parameters

**dropInfo** (ODDragItemIterator \*) - input

A reference to a drag-item iterator that describes the content, as well as the types and values, of the dragged data.

**facet** (ODFacet \*) - input

A reference to a facet where the drop has occurred.

**where** (ODPoint \*) - input

The location of the drop point, expressed in frame coordinates.

**rc** (ODDropResult) - returns

The result of the drop operation. This parameter can return one of the following values:

kODDropCopy

A synchronous copy operation was successful.

kODDropFail

A synchronous drop operation was unsuccessful.

kODDropMove

A synchronous move operation was successful

kODDropUnfinished

An asynchronous drop is in progress (on platforms that support asynchronous drops.)

---

## Drop - Remarks

The copy and move semantics are determined by examining the drag attributes and determining whether to display the Paste As dialog box. If a link is created, your part's Drop method should return kODDropCopy, regardless of the drag attribute.

OpenDoc calls this method when the mouse button is released while the drag point is within a facet that can accept a drop.

Your part's Drop method should examine the part kinds of the dragged data (using the drag-item iterator specified by the *dropInfo* parameter) and determine whether it can accept the dragged data. If your part can accept a drop, this method should incorporate or embed the dropped data and return an appropriate return value indicating the result of the drop operation. If your part cannot accept a drop, this method should return `kODDropFail`.

---

## Drop - Exception Handling

`kODErrDoesNotDrop`

This part does not support drag and drop.

---

## Drop - Override Policy

If you subclass [ODPart](#), you can override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## Drop - Related Methods

### Related Methods

- [ODDragAndDrop::GetDragAttributes](#)
- [ODDragAndDrop::ShowPasteAsDialog](#)
- [ODPart::DropCompleted](#)

---

## Drop - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

[Related Methods](#)

---

## DropCompleted

---

## DropCompleted - Syntax

This method notifies this part that a drop operation, resulting from an asynchronous drag initiated from this part, is complete.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODPart      *destPart;
ODDDropResult dropResult;

DropCompleted(destPart, dropResult);
```

---

## DropCompleted Parameter - destPart

**destPart** (ODPart \*) - input  
A reference to a part where the drop is to be completed.

---

## DropCompleted Parameter - dropResult

**dropResult** (ODDropResult) - input  
The result of the drop operation. This parameter can be set to one of the following values:

kODDDropCopy	A synchronous copy operation was successful.
kODDDropFail	A synchronous drop operation was unsuccessful.
kODDDropMove	A synchronous move operation was successful
kODDDropUnfinished	An asynchronous drop is in progress (on platforms that support asynchronous drops).

---

## DropCompleted - Return Value

None.

---

## DropCompleted - Parameters

**destPart** (ODPart \*) - input  
A reference to a part where the drop is to be completed.

**dropResult** (ODDropResult) - input  
The result of the drop operation. This parameter can be set to one of the following values:

kODDDropCopy

kODDDropFail	A synchronous copy operation was successful.
kODDDropMove	A synchronous drop operation was unsuccessful.
kODDDropUnfinished	A synchronous move operation was successful
	An asynchronous drop is in progress (on platforms that support asynchronous drops).
None.	

-----

## DropCompleted - Remarks

OpenDoc calls this method at the end of a drop operation resulting from an asynchronous drag initiated from this part.

If the drag-and-drop object's [StartDrag](#) method is executed synchronously, that method's return value is the method's drop result.

Your part's DropCompleted method should perform any tasks that your part normally performs when it receives the result of a [StartDrag](#) method, modifying or restoring this part's data based on the drop result.

-----

## DropCompleted - Exception Handling

kODErrDoesNotDrop	This part does not support drag and drop.
-------------------	---

-----

## DropCompleted - Override Policy

If you subclass [ODPart](#), you can override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

-----

## DropCompleted - Related Methods

### Related Methods

- [ODDragAndDrop::StartDrag](#)
- [ODPart::Drop](#)

-----

## DropCompleted - Topics

### Class:

ODPart

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## EditInLinkAttempted

---

## EditInLinkAttempted - Syntax

This method indicates whether this part maintains the destination of a link that includes the embedded frame.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODFrame      *frame;
ODBoolean     rc;

rc = EditInLinkAttempted(frame);
```

---

## EditInLinkAttempted Parameter - frame

**frame** (ODFrame \*) - input  
A reference to an embedded frame, in which the edit was attempted, of this part.

---

## EditInLinkAttempted Return Value - rc

**rc** (ODBoolean) - returns  
A flag indicating whether this part maintains the destination of a link that includes the embedded frame.

kODTrue	This part maintains the destination of a link that includes the embedded frame.
kODFalse	This part does not maintain the destination of a link that includes the embedded frame.

---

## EditInLinkAttempted - Parameters

**frame** (ODFrame \*) - input  
A reference to an embedded frame, in which the edit was attempted, of this part.

**rc** ([ODBoolean](#)) - returns

A flag indicating whether this part maintains the destination of a link that includes the embedded frame.

kODTrue

This part maintains the destination of a link that includes the embedded frame.

kODFalse

This part does not maintain the destination of a link that includes the embedded frame.

-----

## EditInLinkAttempted - Remarks

OpenDoc calls this method to notify this part that an attempt was made to edit content within a frame embedded in the destination of a link maintained by this part. The frame may be embedded at any depth within this part or its embedded parts.

Without making itself active, your part should display an alert box informing the user of the attempted edit to linked content and allow the user to find the source of the link or to break the link. If the user chooses to break the link, your part should change the link status of the embedded frame; the user can then retry the editing operation in the still-active frame.

Your part's EditInLinkAttempted method is not called by most parts. If your part's active frame is within a link destination and the user attempts to edit its content, you call the frame's [EditInLink](#) method instead of this method.

-----

## EditInLinkAttempted - Exception Handling

kODErrDoesNotLink

This part does not support linking.

-----

## EditInLinkAttempted - Override Policy

If you subclass [ODPart](#), you can override this method if you support linking and embedding. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely. This method needs to be implemented only by container parts that support linking.

-----

## EditInLinkAttempted - Related Methods

### Related Methods

- [ODFrame::EditInLink](#)

-----

## EditInLinkAttempted - Topics

### Class:

ODPart

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

---

## EmbeddedFrameSpec

---

### EmbeddedFrameSpec - Syntax

This method creates an object specifier for the specified embedded frame in this part.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODFrame          *embeddedFrame;
ODObjectSpec     *spec;

EmbeddedFrameSpec(embeddedFrame, spec);
```

---

### EmbeddedFrameSpec Parameter - embeddedFrame

**embeddedFrame** (ODFrame \*) - input  
A reference to an embedded frame of this part.

---

### EmbeddedFrameSpec Parameter - spec

**spec** (ODObjectSpec \*) - input  
The object specifier for the frame.

---

### EmbeddedFrameSpec - Return Value

None.

---

### EmbeddedFrameSpec - Parameters

**embeddedFrame** (ODFrame \*) - input  
A reference to an embedded frame of this part.

**spec** (ODOBJECTSpec \*) - input  
The object specifier for the frame.

None.

---

## EmbeddedFrameSpec - Remarks

An object specifier is a designation of a content object within a part. An object specifier is used to determine the target of a semantic event (a message sent to a part or one of its content elements). Object specifiers can be names (for example, "blue rectangle") or logical designations (such as "word 1 of line 2 of the embedded frame 3"). OpenDoc or parts can call this method to create an object specifier for an embedded frame to distinguish it from other frames.

If your part is an embedded part, this method should first obtain the specifier for your part's display frame by calling the EmbeddedFrameSpec method of your part's containing part and then concatenate that returned specifier with the object specifier for the specified embedded frame.

---

## EmbeddedFrameSpec - Exception Handling

kODErrCannotEmbed

This part does not support embedding.

kODErrInvalidFrame

The specified frame is not an embedded frame of this part.

---

## EmbeddedFrameSpec - Override Policy

If you subclass [ODPart](#), you can override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely. This method needs to be implemented only by container parts that support scripting.

---

## EmbeddedFrameSpec - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

---

## EmbeddedFrameUpdated



---

## EmbeddedFrameUpdated - Syntax

This method updates any of this part's link-source objects that are affected by a change to the specified embedded frame.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODFrame      *frame;
ODUpdateID    change;

EmbeddedFrameUpdated(frame, change);
```

---

## EmbeddedFrameUpdated Parameter - frame

**frame** (ODFrame \*) - input  
A reference to an embedded frame whose content has changed.

---

## EmbeddedFrameUpdated Parameter - change

**change** (ODUpdateID) - input  
The updated ID associated with the frame.

---

## EmbeddedFrameUpdated - Return Value

None.

---

## EmbeddedFrameUpdated - Parameters

**frame** (ODFrame \*) - input  
A reference to an embedded frame whose content has changed.

**change** (ODUpdateID) - input  
The updated ID associated with the frame.

None.

---

## EmbeddedFrameUpdated - Remarks

An embedded frame's [ContentUpdated](#) method calls this method when its content changes. Your part's `EmbeddedFrameUpdated` method is called recursively for all containing parts in the frame hierarchy through the root part of the window displaying the embedded frame whose content has changed; therefore, this part is not responsible for notifying its containing part of the change. This part may ignore the notification if it is uninterested in changes in embedded content.

If the embedded frame is involved in a link source with your part, your part's `EmbeddedFrameUpdated` method should update the link-source object with the new data.

Your part's `EmbeddedFrameUpdated` method may be called multiple times with the same update ID; however, your part should wait a certain length of time (perhaps, a second) before updating its display so that subsequent calls to your method (with the same update ID) do not result in multiple updates for the same change.

---

## EmbeddedFrameUpdated - Exception Handling

`kODErrCannotEmbed`

This part does not support embedding.

---

## EmbeddedFrameUpdated - Override Policy

If you subclass [ODPart](#), you can override this method if you want your part to do something special in response to the `EmbeddedFrameUpdated` notification. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely. This method needs to be implemented only by container parts.

---

## EmbeddedFrameUpdated - Related Methods

### Related Methods

- [ODFrame::ContentUpdated](#)

---

## EmbeddedFrameUpdated - Topics

### Class:

`ODPart`

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

[Related Methods](#)

---

# ExternalizeKinds

---

## ExternalizeKinds - Syntax

This method should write to storage a representation for each part kind within the specified kind list that this part supports.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODTypeList      *kindset;

ExternalizeKinds(kindset);
```

---

## ExternalizeKinds Parameter - kindset

**kindset** (ODTypeList \*) - input  
A reference to a type list specifying a set of part kinds.

---

## ExternalizeKinds - Return Value

None.

---

## ExternalizeKinds - Parameters

**kindset** (ODTypeList \*) - input  
A reference to a type list specifying a set of part kinds.

None.

---

## ExternalizeKinds - Remarks

OpenDoc calls this method. OpenDoc does not ensure that your part supports a subset of the part kinds in the specified kind list; you should write a representation for as many of the part kinds in the specified kind list that your part supports.

A part's ExternalizeKinds method does not specify anything about the ordering of those kinds in the content property of your part. Make sure that the ordering of the values in your contents property reflects your part editor's fidelity order.

---

## ExternalizeKinds - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## ExternalizeKinds - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

---

## FacetAdded

---

## FacetAdded - Syntax

This method should notify this part that a facet has been added to one of its display frames.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>
```

```
ODFacet      *facet;
```

```
FacetAdded(facet);
```

---

## FacetAdded Parameter - facet

**facet** (ODFacet \*) - input

A reference to a facet that has been added.

---

## FacetAdded - Return Value

None.

---

## FacetAdded - Parameters

**facet** (ODFacet \*) - input

A reference to a facet that has been added.

None.

---

## FacetAdded - Remarks

OpenDoc calls this method when your part's containing part adds a facet to one of your part's display frames. If your part is the root part in a window, OpenDoc calls this method when the window is opened.

Your part's FacetAdded method should perform any actions necessary to handle the addition of the new facet to one of your part's display frames. Some actions depend on the nature and implementation of your part itself, but others are standard. Standard actions include creating facets for all visible embedded frames within the area of the added facet, storing appropriate part-information data in the facet, examining the facet's canvas to make sure your part editor understands how to draw on that canvas, and creating an offscreen canvas, if the facet needs one.

If your part does not support asynchronous display and is not a container part, it is not necessary for your part to do anything except possibly validate that it can draw to its canvas.

---

## FacetAdded - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## FacetAdded - Related Methods

### Related Methods

- [ODFrame::FacetAdded](#)
  - [ODPart::FacetRemoved](#)
- 

## FacetAdded - Topics

**Class:**

ODPart

Select an item:

[Syntax](#)[Parameters](#)[Returns](#)[Remarks](#)[Override Policy](#)[Related Methods](#)

---

## FacetRemoved

---

## FacetRemoved - Syntax

The method should notify this part that a facet has been removed from one of its display frames.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODFacet      *facet;

FacetRemoved(facet);
```

---

## FacetRemoved Parameter - facet

**facet** (ODFacet \*) - input

A reference to a facet that has been removed.

---

## FacetRemoved - Return Value

None.

---

## FacetRemoved - Parameters

**facet** (ODFacet \*) - input

A reference to a facet that has been removed.

None.

-----

## FacetRemoved - Remarks

OpenDoc calls this method when your part's containing part removes a facet from one of your part's display frames. If your part is the root part in a window, OpenDoc also calls this method when the window is closed.

Your part's FacetRemoved method should perform any actions necessary to handle the removal of the facet. In general, the actions performed by this method should reverse those performed by the [FacetAdded](#) method. If your part does not support asynchronous display and is not a container part, its response to this call is minimal.

-----

## FacetRemoved - Exception Handling

kODErrInvalidFacet

The specified facet is not a facet of this part.

-----

## FacetRemoved - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

-----

## FacetRemoved - Related Methods

### Related Methods

- [ODFrame::FacetRemoved](#)
- [ODPart::FacetAdded](#)

-----

## FacetRemoved - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

[Related Methods](#)

---

# FocusAcquired

---

## FocusAcquired - Syntax

The method should notify this part that one of its display frames has acquired the specified focus.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODTypeToken    focus;
ODFrame        *ownerFrame;

FocusAcquired(focus, ownerFrame);
```

---

## FocusAcquired Parameter - focus

**focus** (ODTypeToken) - input

A tokenized string representing the focus type to be acquired.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus	The frame with the clipboard focus has access to the clipboard.
kODKeyFocus	The frame with keyboard focus receives keyboard events (excluding page-movement key events).
kODMenuFocus	The frame with menu focus receives menu events.
kODModalFocus	A frame with modal focus is notifying other frames that it is the only currently modal frame.
kODMouseFocus	The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.
kODScrollingFocus	The frame with the scrolling focus receives page-movement key events (such as <b>PageUp</b> ). This frame does not need to own keyboard focus.
kODSelectionFocus	The frame with the selection focus receives modified mouse-click events ( <b>Shift</b> +click and <b>Ctrl</b> +click). OpenDoc draws the active frame border around all facets of this frame.

---

## FocusAcquired Parameter - ownerFrame

**ownerFrame** (ODFrame \*) - input

A reference to the display frame that has acquired the focus.

---



# FocusAcquired - Return Value

None.

---

## FocusAcquired - Parameters

**focus** ([ODTypeToken](#)) - input

A tokenized string representing the focus type to be acquired.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

**kODClipboardFocus**

The frame with the [clipboard](#) focus has access to the clipboard.

**kODKeyFocus**

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

**kODMenuFocus**

The frame with menu focus receives menu events.

**kODModalFocus**

A frame with modal focus is notifying other frames that it is the only currently modal frame.

**kODMouseFocus**

The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

**kODScrollingFocus**

The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

**kODSelectionFocus**

The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

**ownerFrame** ([ODFrame](#) \*) - input

A reference to the display frame that has acquired the focus.

None.

---

## FocusAcquired - Remarks

OpenDoc calls this method to notify this part that the ownership of a focus has been transferred to it. For example, if a containing part uses the arbitrator's [TransferFocus](#) method to transfer a focus directly from one embedded part to another, the focus module calls the destination part's [FocusAcquired](#) method.

Your part's [FocusAcquired](#) method is not called when the part requests a focus or focus set using the arbitrator's [RequestFocus](#) and [RequestFocusSet](#) methods.

Your part's [FocusAcquired](#) method should perform any actions necessary to indicate that it has acquired the focus. For example, acquiring the selection focus might cause your part to highlight its selection.

---

## FocusAcquired - Exception Handling

## FocusAcquired - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## FocusAcquired - Related Methods

### Related Methods

- [ODPart::FocusLost](#)
  - [ODSession::Tokenize](#)
- 

## FocusAcquired - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## FocusLost

---

## FocusLost - Syntax

The method should notify this part that one of its display frames has lost the specified focus.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODTypeToken    focus;
ODFrame        *ownerFrame;

FocusLost(focus, ownerFrame);
```

---

## FocusLost Parameter - focus

**focus** (ODTypeToken) - input

A tokenized string representing the focus type that was lost.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the clipboard focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus

The frame with menu focus receives menu events.

kODModalFocus

A frame with modal focus is notifying other frames that it is the only currently modal frame.

kODMouseFocus

The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.

kODScrollingFocus

The frame with the scrolling focus receives page-movement key events (such as **PageUp**). This frame does not need to own keyboard focus.

kODSelectionFocus

The frame with the selection focus receives modified mouse-click events (**Shift**+click and **Ctrl**+click). OpenDoc draws the active frame border around all facets of this frame.

---

## FocusLost Parameter - ownerFrame

**ownerFrame** (ODFrame \*) - input

A reference to a display frame that has lost the focus.

---

## FocusLost - Return Value

None.

---

## FocusLost - Parameters

**focus** (ODTypeToken) - input

A tokenized string representing the focus type that was lost.

This parameter must be the tokenized form of one of the following focus constants or the tokenized form of a part-specific focus type. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODClipboardFocus

The frame with the clipboard focus has access to the clipboard.

kODKeyFocus

The frame with keyboard focus receives keyboard events (excluding page-movement key events).

kODMenuFocus	The frame with menu focus receives menu events.
kODModalFocus	A frame with modal focus is notifying other frames that it is the only currently modal frame.
kODMouseFocus	The frame with mouse focus receives all mouse events independent of the facet in which the mouse pointer is located.
kODScrollingFocus	The frame with the scrolling focus receives page-movement key events (such as <b>PageUp</b> ). This frame does not need to own keyboard focus.
kODSelectionFocus	The frame with the selection focus receives modified mouse-click events ( <b>Shift</b> +click and <b>Ctrl</b> +click). OpenDoc draws the active frame border around all facets of this frame.

**ownerFrame** (ODFrame \*) - input

A reference to a display frame that has lost the focus.

None.

-----

## FocusLost - Remarks

OpenDoc calls this method to notify this part that the ownership of a focus has been unilaterally removed. For example, a focus module might detect that some physical hardware connection has ben broken. Or if a containing part uses the arbitrator's [TransferFocus](#) method to transfer a focus directly from one embedded part to another, the focuss module calls the source part's FocusLost method.

Your part's FocusLost method is not called when this part loses a focus because another part has requested it-that is, using the arbitrator's [RequestFocus](#) and [RequestFocusSet](#) methods. In this case, the part's [CommitRelinquishFocus](#) method is called if the part agrees to relinquish the focus.

Your part's FocusLost method should perform any actions necessary to indicate that it has lost the focus (for example, closing connections and removing highlighting). Your part should avoid inappropriate behavior after it has lost the focus. For example, your part should not attempt to adjust menus or display a menu bar when your part does not have the menu focus.

-----

## FocusLost - Exception Handling

kODErrInvalidFrame

The specified frame is not a display frame of this part.

-----

## FocusLost - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

-----

## FocusLost - Related Methods

### Related Methods

- [ODPart::AbortRelinquishFocus](#)
- [ODPart::BeginRelinquishFocus](#)
- [ODPart::CommitRelinquishFocus](#)
- [ODPart::FocusAcquired](#)
- [ODSession::Tokenize](#)

---

## FocusLost - Topics

### Class:

ODPart

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

[Related Methods](#)

---

## FrameShapeChanged

---

### FrameShapeChanged - Syntax

This method should notify this part that the frame shape of one of its display frames has changed.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODFrame      *frame;

FrameShapeChanged(frame);
```

---

### FrameShapeChanged Parameter - frame

**frame** (ODFrame \*) - input

A reference to a display frame that was reshaped.

---

### FrameShapeChanged - Return Value

None.

---

## FrameShapeChanged - Parameters

**frame** (ODFrame \*) - input  
A reference to a display frame that was reshaped.

None.

---

## FrameShapeChanged - Remarks

OpenDoc calls this method when it or this part's containing part changes the frame shape of one of this part's display frames.

This part should perform any actions necessary to respond to the new shape. For example, your part should change its content layout, change its used shape, or resize its embedded frames.

Your part also has the option of requesting a different frame shape using its display frame's [RequestFrameShape](#) method, though it must be able to accept the shape it is given. If the size of the frame is insufficient, your part may ask its containing part for additional frames by calling its containing part's [RequestEmbeddedFrame](#) method.

---

## FrameShapeChanged - Exception Handling

kODErrInvalidFrame

The specified frame is not a display frame of this part.

---

## FrameShapeChanged - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## FrameShapeChanged - Related Methods

### Related Methods

- [ODFrame::ChangeFrameShape](#)
  - [ODPart::RequestEmbeddedFrame](#)
  - [ODPart::RequestFrameShape](#)
- 

## FrameShapeChanged - Topics

**Class:**

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

[Related Methods](#)

---

## FulfillPromise

---

### FulfillPromise - Syntax

This method fulfills a promise by providing the content data the promise represents.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitView      *promiseSUIView;

FulfillPromise(promiseSUIView);
```

---

### FulfillPromise Parameter - promiseSUIView

**promiseSUIView** (ODStorageUnitView \*) - input

A reference to a storage-unit view object that contains the promise. This is the same value created by the storage unit's [SetPromiseValue](#) method.

---

### FulfillPromise - Return Value

None.

---

### FulfillPromise - Parameters

**promiseSUIView** (ODStorageUnitView \*) - input

A reference to a storage-unit view object that contains the promise. This is the same value created by the storage unit's

[SetPromiseValue](#) method.

None.

-----

## FulfillPromise - Remarks

A promise is a specification of data to be transferred at a future time. For data interchange using drag-and-drop or the clipboard, the part transferring data should usually delay the actual data transfer, instead providing a promise that is fulfilled only when the transfer is ultimately required.

If a data transfer involves a very large amount of data, your part can choose to put out a promise instead of actually writing the data to a storage unit. Your part's FulfillPromise method can then write the actual data only if and when a transfer to another part occurs. When cloning in the FulfillPromise method, the clone kind should be the same as the one used when the promise was written. OpenDoc calls this method when a promise must be fulfilled.

-----

## FulfillPromise - Exception Handling

kODErrNoPromises

This part did not make the promise.

-----

## FulfillPromise - Override Policy

If you subclass [ODPart](#), you can override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

-----

## FulfillPromise - Related Methods

### Related Methods

- [ODStorageUnit::SetPromiseValue](#)

-----

## FulfillPromise - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Exception Handling](#)  
[Related Methods](#)



---

# GeometryChanged

---

## GeometryChanged - Syntax

This method should notify this part that the clip shape or external transform (or both) of one of this part's facets has changed.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODFacet      *facet;
ODBoolean     clipShapeChanged;
ODBoolean     externalTransformChanged;

GeometryChanged(facet, clipShapeChanged, externalTransformChanged);
```

---

## GeometryChanged Parameter - facet

**facet** (ODFacet \*) - input  
A reference to a facet with the changed geometry.

---

## GeometryChanged Parameter - clipShapeChanged

**clipShapeChanged** (ODBoolean) - input  
A flag indicating whether the clip shape has changed.

kODTrue	The clip shape has changed.
kODFalse	The clip shape has not changed.

---

## GeometryChanged Parameter - externalTransformChanged

**externalTransformChanged** (ODBoolean) - input  
A flag indicating whether the external transform has changed.

kODTrue	The external transform has changed.
kODFalse	The external transform has not changed.

---

## GeometryChanged - Return Value

None.

---

## GeometryChanged - Parameters

**facet** (ODFacet \*) - input

A reference to a facet with the changed geometry.

**clipShapeChanged** (ODBoolean) - input

A flag indicating whether the clip shape has changed.

kODTrue

The clip shape has changed.

kODFalse

The clip shape has not changed.

**externalTransformChanged** (ODBoolean) - input

A flag indicating whether the external transform has changed.

kODTrue

The external transform has changed.

kODFalse

The external transform has not changed.

None.

---

## GeometryChanged - Remarks

OpenDoc calls this method when this part's facet changes its clip shape, external transform, or both, or when this part's facet is repositioned or clipped differently.

Your part's GeometryChanged method should use the new clip shape for display. If your part only displays in response to update events, it does not need to do anything but check the clip shape each time it draws. If your part supports asynchronous display (for example, clocks and movies), it must notice the new clip shape and limit its display accordingly.

---

## GeometryChanged - Exception Handling

kODErrInvalidFacet

The specified facet is not a facet of this part.

---

## GeometryChanged - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must not call its inherited method; that is, your override method

must implement this method's functionality completely.

---

## GeometryChanged - Related Methods

### Related Methods

- [ODFacet::ChangeGeometry](#)
- 

## GeometryChanged - Topics

### Class:

ODPart

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## GetPrintResolution

---

## GetPrintResolution - Syntax

This method should return the minimum desired resolution required for printing the content of the specified display frame.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODFrame      *frame;
ODULong      rc;

rc = GetPrintResolution(frame);
```

---

## GetPrintResolution Parameter - frame

**frame** (ODFrame \*) - input

A reference to a display frame for which the resolution is needed.

---

# GetPrintResolution Return Value - rc

**rc** ([ODULong](#)) - returns  
The minimum desired resolution, expressed in dots per inch.

---

## GetPrintResolution - Parameters

**frame** ([ODFrame](#) \*) - input  
A reference to a display frame for which the resolution is needed.

**rc** ([ODULong](#)) - returns  
The minimum desired resolution, expressed in dots per inch.

---

## GetPrintResolution - Remarks

The root part calls this method when creating the document's print job to guarantee that it can display the highest-resolution part.

---

## GetPrintResolution - Exception Handling

[kODErrInvalidFrame](#)                      The specified frame is not a display frame of this part.

---

## GetPrintResolution - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## GetPrintResolution - Topics

**Class:**  
[ODPart](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Exception Handling](#)

---

# GetRealPart

---

## GetRealPart - Syntax

This method returns a reference to a part object encapsulated by the part wrapper.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODPart      *rc;

rc = GetRealPart();
```

---

## GetRealPart Return Value - rc

**rc** (ODPart \*) - returns  
A reference to a part object encapsulated by the part wrapper.

---

## GetRealPart - Parameters

**rc** (ODPart \*) - returns  
A reference to a part object encapsulated by the part wrapper.

---

## GetRealPart - Remarks

Use of this method must conform to the following constraints:

- Only one client at a time may have access to the actual part-that is, clients should not hold on to a reference to the actual part object for longer than absolutely necessary; never store a reference to the part object and do not pass it as a parameter to other objects.
- Your part's GetRealPart method increments the reference count of the returned part. When you have finished using the actual part, you should call its [ReleaseRealPart](#) method. If another client attempts to get the actual part before the [ReleaseRealPart](#) method is called, an exception is generated.

Your part's GetRealPart method is almost never called unless you really need to access the actual part.

---

## GetRealPart - Exception Handling

kODErrPartNotWrapper

The method was called on an actual part, not a part wrapper.

---

## GetRealPart - Override Policy

If you subclass [ODPart](#), you must not override this method.

---

## GetRealPart - Related Methods

### Related Methods

- [ODPart::IsRealPart](#)
  - [ODPart::ReleaseRealPart](#)
- 

## GetRealPart - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

[Related Methods](#)

---

## HandleEvent

---

## HandleEvent - Syntax

This method should attempt to handle the specified user event.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>
```

```
ODEEventData    *event;
ODFrame         *frame;
ODFacet         *facet;
ODEventInfo     *eventInfo;
```

```
ODBoolean rc;

rc = HandleEvent(event, frame, facet, eventInfo);
```

---

## HandleEvent Parameter - event

**event** (ODEventData \*) - in/out  
A platform-specific structure representing an event. On return, the fields of the structure may have been modified.

---

## HandleEvent Parameter - frame

**frame** (ODFrame \*) - input  
A reference to a display frame in which the event occurred.

---

## HandleEvent Parameter - facet

**facet** (ODFacet \*) - input  
A reference to a facet in which the event occurred or KODNULL for events not based on geometry (such as keyboard events) or events outside a modal focus.

---

## HandleEvent Parameter - eventInfo

**eventInfo** (ODEventInfo \*) - in/out  
A platform-specific structure that contains additional event information. On return, the relative fields of the structure are filled in if the event was handled.

---

## HandleEvent Return Value - rc

**rc** (ODBoolean) - returns  
A flag indicating whether this part handled the event.

KODTrue	This part handled the event.
KODFalse	This part was not able to handle the event.

---

# HandleEvent - Parameters

**event** ([ODEEventData \\*](#)) - in/out

A platform-specific structure representing an event. On return, the fields of the structure may have been modified.

**frame** ([ODFrame \\*](#)) - input

A reference to a display frame in which the event occurred.

**facet** ([ODFacet \\*](#)) - input

A reference to a facet in which the event occurred or `kODNULL` for events not based on geometry (such as keyboard events) or events outside a modal focus.

**eventInfo** ([ODEEventInfo \\*](#)) - in/out

A platform-specific structure that contains additional event information. On return, the relative fields of the structure are filled in if the event was handled.

**rc** ([ODBoolean](#)) - returns

A flag indicating whether this part handled the event.

`kODTrue`

This part handled the event.

`kODFalse`

This part was not able to handle the event.

-----

## HandleEvent - Remarks

OpenDoc calls this method to pass user events to this part.

If this part has set the *doesPropagateEvents* flag for any of its embedded frames (by calling the embedded frame's [SetPropagateEvents](#) method), this part then receives any event not handled by an embedded frame in addition to its own events.

-----

## HandleEvent - Exception Handling

`kODErrInvalidFacet`  
`kODErrInvalidFrame`

The specified facet is not a facet of this part.  
The specified frame is not a display frame of this part.

-----

## HandleEvent - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

-----

## HandleEvent - Related Methods

### Related Methods

- [ODFrame::SetPropagateEvents](#)



---

# HandleEvent - Topics

## Class:

ODPart

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

[Related Methods](#)

---

## HighlightChanged

---

### HighlightChanged - Syntax

This method should update the highlight state of the specified facet of this part.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODFacet      *facet;

HighlightChanged(facet);
```

---

### HighlightChanged Parameter - facet

**facet** (ODFacet \*) - input

A reference to a facet of the part.

---

### HighlightChanged - Return Value

None.

---

### HighlightChanged - Parameters

**facet** (ODFacet \*) - input  
A reference to a facet of the part.

None.

---

## HighlightChanged - Remarks

OpenDoc calls this method when the highlight state of one of your part's facets changes. This allows your part to draw its content consistently with the content highlighting of your part's containing part.

Your part's HighlightChanged method should adjust your part's presentation in the facet to its new highlight state. The new state is found by calling the facet's [GetHighlight](#) method.

---

## HighlightChanged - Exception Handling

kODErrInvalidFacet

The specified facet is not a facet of this part.

---

## HighlightChanged - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## HighlightChanged - Related Methods

### Related Methods

- [ODFacet::ChangeHighlight](#)
  - [ODFacet::GetHighlight](#)
- 

## HighlightChanged - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

---

# InitPart

---

## InitPart - Syntax

This method should initialize this part object.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODStorageUnit    *storageUnit;
ODPart           *partWrapper;

InitPart(storageUnit, partWrapper);
```

---

## InitPart Parameter - storageUnit

**storageUnit** (ODStorageUnit \*) - input

A reference to the empty storage unit to be used by this part as its primary persistent storage.

---

## InitPart Parameter - partWrapper

**partWrapper** (ODPart \*) - input

A reference to the part wrapper representing this part.

---

## InitPart - Return Value

None.

---

## InitPart - Parameters

**storageUnit** (ODStorageUnit \*) - input

A reference to the empty storage unit to be used by this part as its primary persistent storage.

**partWrapper** (ODPart \*) - input

A reference to the part wrapper representing this part.

None.

---

## InitPart - Remarks

OpenDoc calls this method only once for the persistent lifetime of this part, when it is first created and has no stored data to read. After this method executes successfully, this part object is initialized and ready for use.

Your part's InitPart method, rather than its somlInit method, should handle any initialization code that can potentially fail. Your part's InitPart method may attempt to allocate extra memory for your part instance, get resources that your part might need, or set up your part's persistent storage.

The inherited InitPart method creates and stores the kODPropCreateDate, kODPropModDate, and kODPropModUser properties; your part's storage unit automatically maintains the kODPropModDate and kODPropModUser properties.

---

## InitPart - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must call its inherited method at the beginning of your implementation.

---

## InitPart - Topics

### Class:

ODPart

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)

---

## InitPartFromStorage

---

## InitPartFromStorage - Syntax

This method should initialize this part object from its stored data.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODStorageUnit      *storageUnit;
ODPart              *partWrapper;

InitPartFromStorage(storageUnit, partWrapper);
```

-----

## InitPartFromStorage Parameter - storageUnit

**storageUnit** (ODStorageUnit \*) - input

A reference to a specified storage unit from which this part should read its persistent state.

-----

## InitPartFromStorage Parameter - partWrapper

**partWrapper** (ODPart \*) - input

A reference to a part wrapper representing this part.

-----

## InitPartFromStorage - Return Value

None.

-----

## InitPartFromStorage - Parameters

**storageUnit** (ODStorageUnit \*) - input

A reference to a specified storage unit from which this part should read its persistent state.

**partWrapper** (ODPart \*) - input

A reference to a part wrapper representing this part.

None.

-----

## InitPartFromStorage - Remarks

Whenever a document containing this part is opened, or if this part is added to a document by means of data transfer, the part must be instantiated and read into memory. OpenDoc calls this method to initialize the run-time part object from persistent storage.

Your part's `InitPartFromStorage` method is similar to its [InitPart](#) method, except that it reads in data. OpenDoc passes a storage unit to your part, from which your part reads itself. Your part is responsible for reading the storage unit and preparing itself to receive other messages. Your part should retrieve, from its content property, the value that represents the data stream to be read.

---

## InitPartFromStorage - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must call its inherited method at the beginning of your implementation.

---

## InitPartFromStorage - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

---

## IsRealPart

---

## IsRealPart - Syntax

This method indicates whether this part is an actual part (as opposed to a part wrapper).

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODBoolean    isRealPart;

isRealPart = IsRealPart();
```

---

## IsRealPart Return Value - isRealPart

**isRealPart** ([ODBoolean](#)) - returns

A flag indicating whether this part is a part or part wrapper.

kODTrue

This part is an actual part.

kODFalse

This part is a part wrapper.

---

## IsRealPart - Parameters

**isRealPart** ([ODBoolean](#)) - returns

A flag indicating whether this part is a part or part wrapper.

kODTrue

This part is an actual part.

kODFalse

This part is a part wrapper.

---

## IsRealPart - Remarks

Your part's IsRealPart method is almost never called unless you really need to access the actual part.

---

## IsRealPart - Override Policy

If you subclass [ODPart](#), you must not override this method.

---

## IsRealPart - Related Methods

### Related Methods

- [ODPart::GetRealPart](#)
  - [ODPart::ReleaseRealPart](#)
- 

## IsRealPart - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Related Methods](#)

---

## LinkBroken (OS/2)

---

## LinkBroken (OS/2) - Syntax

This method is used by the *Link Source* pages of the Properties notebook to indicate that the use has broken a link from the link source end.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODLink      *link;

LinkBroken(link);
```

---

## LinkBroken (OS/2) Return Value - link

**link** (ODLink \*) - returns  
The link object to break a link with.

---

## LinkBroken (OS/2) - Return Value

None.

---

## LinkBroken (OS/2) - Parameters

**link** (ODLink \*) - returns  
The link object to break a link with.

None.

---

## LinkBroken (OS/2) - Remarks

There is no default action in this method except to raise an exception `KODErrSubClassResponsibility`.

---

## LinkBroken (OS/2) - Exception Handling



---

## LinkBroken (OS/2) - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must perform any part-specific cleanup and should call the link's [Release](#) method.

---

## LinkBroken (OS/2) - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

---

## LinkStatusChanged

---

## LinkStatusChanged - Syntax

This method should notify this part that the link status of one of its display frames has changed.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODFrame      *frame;

LinkStatusChanged(frame);
```

---

## LinkStatusChanged Parameter - frame

**frame** (ODFrame \*) - input

A reference to a display frame whose link status has changed.

---

## LinkStatusChanged - Return Value

None.

---

## LinkStatusChanged - Parameters

**frame** (ODFrame \*) - input  
A reference to a display frame whose link status has changed.

None.

---

## LinkStatusChanged - Remarks

OpenDoc calls this method. Your part's LinkStatusChanged method should call the [ChangeLinkStatus](#) method of each embedded frame affected by the display frame change, assigning it the same link status as the display frame.

If the value kODNotInLink was assigned in the [ChangeLinkStatus](#) method, your part's LinkStatusChanged method can still get the status of this part's display frame (its containing frame).

---

## LinkStatusChanged - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## LinkStatusChanged - Related Methods

### Related Methods

- [ODFrame::ChangeLinkStatus](#)
- 

## LinkStatusChanged - Topics

**Class:**  
[ODPart](#)

Select an item:  
[Syntax](#)

[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Related Methods](#)

---

## LinkUpdated

---

## LinkUpdated - Syntax

This method should replace the content at each destination of a link with new content from an updated link object.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODLink      *updatedLink;
ODUpdateID  change;

LinkUpdated(updatedLink, change);
```

---

## LinkUpdated Parameter - updatedLink

**updatedLink** (ODLink \*) - input  
A reference to a link that has changed.

---

## LinkUpdated Parameter - change

**change** (ODUpdateID) - input  
The update ID associated with the link; an identifier for a particular version of link-source data.

---

## LinkUpdated - Return Value

None.

---

## LinkUpdated - Parameters

**updatedLink** (ODLink \*) - input  
A reference to a link that has changed.

**change** (ODUpdateID) - input  
The update ID associated with the link; an identifier for a particular version of link-source data.

None.

-----

## LinkUpdated - Remarks

Each link object maintains a registry of dependent parts. If this part is registered as a dependent of the link source (by having called the link's [RegisterDependent](#) method), OpenDoc calls this method automatically when the link destination object changes.

Your part's `LinkUpdated` method should retrieve the data from the link, and incorporate or embed that data into your part at the link's destination, thereby replacing any previous content of the link.

It is not always necessary to update link destinations immediately. For example, if the destination has scrolled offscreen but is registered as a dependent of the link, updating does not need to occur until the destination scrolls back into view. Your part editor can, if desired, perform link updating as a background task.

-----

## LinkUpdated - Exception Handling

kODErrDoesNotLink	The specified link is not a link of this part.
-------------------	--

-----

## LinkUpdated - Override Policy

If you subclass [ODPart](#), you can override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

-----

## LinkUpdated - Related Methods

### Related Methods

- [ODLink::RegisterDependent](#)

-----

## LinkUpdated - Topics

**Class:**  
ODPart

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Exception Handling](#)  
[Related Methods](#)

---

# Open

---

## Open - Syntax

This method should create or activate a window in which a frame of this part is the root frame.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODFrame      *frame;
ODID          rc;

rc = Open(frame);
```

---

## Open Parameter - frame

**frame** (ODFrame \*) - input

A reference to a frame that is being opened into a window or KODNULL if the frame does not exist.

---

## Open Return Value - rc

**rc** (ODID) - returns

The ID associated with the window.

---

## Open - Parameters

**frame** (ODFrame \*) - input

A reference to a frame that is being opened into a window or KODNULL if the frame does not exist.

**rc** (ODID) - returns

The ID associated with the window.

---

## Open - Remarks

Your part is always responsible for creating windows in which it is the root part, even when OpenDoc is opening a saved draft. Your part's Open method is called in these circumstances:

- When this part is initially created from stationery—meaning that it has no previously stored frame or window information—OpenDoc calls this method and passes KODNULL for the *frame* parameter.
- When this part is an embedded part whose frame is selected, and the user chooses the **Open Selection** command from the **Document** menu, this part's containing part calls this part's Open method and passes a reference to the selected frame in the *frame* parameter.
- When this part is the root part of a document being opened, OpenDoc calls this method and passes a reference to the root frame in the *frame* parameter. The root frame is annotated with a storage unit containing window information; if the root frame has no window information, this part should instead open a default window.

For more information on opening windows, see the chapter on windows and menus in the *OpenDoc Programming Guide*.

---

## Open - Exception Handling

KODerrInvalidFrame

The specified frame is not a display frame of this part.

---

## Open - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## Open - Related Methods

### Related Methods

- [ODWindowState::Internalize](#)
- 

## Open - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

---

## PresentationChanged

---

## PresentationChanged - Syntax

This method should notify this part that the presentation of one of its display frames has changed.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODFrame      *frame;

PresentationChanged(frame);
```

---

## PresentationChanged Parameter - frame

**frame** (ODFrame \*) - input  
A reference to a display frame for this part.

---

## PresentationChanged - Return Value

None.

---

## PresentationChanged - Parameters

**frame** (ODFrame \*) - input  
A reference to a display frame for this part.

None.

---

## PresentationChanged - Remarks

OpenDoc calls this method when this part's display frame changes its presentation.

Your part's `PresentationChanged` method should examine the new presentation using its frame's [GetPresentation](#) method. It should then display itself in the specified display frame according to the indicated presentation. If your part does not support the requested presentation, it should instead pick a presentation that it can support and then call its frame's [SetPresentation](#) method to update the presentation in the frame.

---

## PresentationChanged - Exception Handling

`kODErrInvalidFrame`

The specified frame is a display frame of this part.

---

## PresentationChanged - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## PresentationChanged - Related Methods

### Related Methods

- [ODFrame::ChangePresentation](#)
- [ODFrame::SetPresentation](#)

---

## PresentationChanged - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## ReadActionState

---

## ReadActionState - Syntax



This method reads the undo action data from the specified storage-unit view object.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODStorageUnitView      *storageUnitView;
ODActionData           rc;

rc = ReadActionState(storageUnitView);
```

---

## ReadActionState Parameter - storageUnitView

**storageUnitView** (ODStorageUnitView \*) - input  
A reference to a storage-unit view object that contains the action data.

---

## ReadActionState Return Value - rc

**rc** (ODActionData) - returns  
A byte array whose buffer contains the data previously logged by this part to allow it to undo the action.

---

## ReadActionState - Parameters

**storageUnitView** (ODStorageUnitView \*) - input  
A reference to a storage-unit view object that contains the action data.

**rc** (ODActionData) - returns  
A byte array whose buffer contains the data previously logged by this part to allow it to undo the action.

---

## ReadActionState - Remarks

OpenDoc calls this method when it reads the undo action data from persistent storage. Your part's ReadActionState method will not be called in OpenDoc 1.0 due to the lack of a persistent undo model; it is provided here for compatibility with future versions of OpenDoc.

---

## ReadActionState - Exception Handling

kODErrDoesNotUndo	The specified action is not an undo action of this part.
kODErrOutOfMemory	There is not enough memory to read the data.

---

## ReadActionState - Override Policy

If you subclass [ODPart](#), you can override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## ReadActionState - Related Methods

### Related Methods

- [ODPart::DisposeActionState](#)
  - [ODPart::WriteActionState](#)
- 

## ReadActionState - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

[Related Methods](#)

---

## ReadPartInfo

---

## ReadPartInfo - Syntax

This method should read the part-information data for a display frame of this part from the specified storage-unit view object.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODFrame          *frame;
ODStorageUnitView *storageUnitView;
ODInfoType       rc;

rc = ReadPartInfo(frame, storageUnitView);
```

---

## ReadPartInfo Parameter - frame

**frame** (ODFrame \*) - input  
A reference to a display frame for this part.

---

## ReadPartInfo Parameter - storageUnitView

**storageUnitView** (ODStorageUnitView \*) - input  
A reference to a storage-unit view object that is focused to the frame's part-information property, but not to any value within that property.

---

## ReadPartInfo Return Value - rc

**rc** (ODInfoType) - returns  
The information stored in the frame's part information.

---

## ReadPartInfo - Parameters

**frame** (ODFrame \*) - input  
A reference to a display frame for this part.

**storageUnitView** (ODStorageUnitView \*) - input  
A reference to a storage-unit view object that is focused to the frame's part-information property, but not to any value within that property.

**rc** (ODInfoType) - returns  
The information stored in the frame's part information.

---

## ReadPartInfo - Remarks

When a document is reopened, OpenDoc may ask this part to read a display frame's part-information data from a particular storage unit back into memory. OpenDoc calls this method when it reads in the part-information data for a display frame of this part.

Your part's ReadPartInfo method should get the storage unit associated with the specified storage-unit view and focus it to the values in the frame's part-information property as necessary to read in the formats it needs.

Your part's ReadPartInfo method should read the data from the storage unit and place it in a block of memory, if necessary. (Your part must first allocate the block of memory.) In simple cases, part-information data could be simple data, such as an integer, which does not require a block of memory. Your part's ReadPartInfo method should then return the memory block to the frame for storage so that your part can access it later.

---

# ReadPartInfo - Exception Handling

kODErrInvalidFrame  
kODErrOutOfMemory

The specified frame is not a display frame of this part.  
There is not enough memory to allocate the part-information data.

---

## ReadPartInfo - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## ReadPartInfo - Related Methods

### Related Methods

- [ODPart::ClonePartInfo](#)
  - [ODPart::WritePartInfo](#)
- 

## ReadPartInfo - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## RedoAction

---

## RedoAction - Syntax

This method redoes the specified action.

```
#define INCL_ODPART
```

```
#define INCL_ODAPI
#include <os2.h>

ODActionData      *actionState;

RedoAction(actionState);
```

-----

## RedoAction Parameter - actionState

**actionState** (ODActionData \*) - input

A byte array whose buffer contains the data previously logged by this part to allow it to redo the action.

-----

## RedoAction - Return Value

None.

-----

## RedoAction - Parameters

**actionState** (ODActionData \*) - input

A byte array whose buffer contains the data previously logged by this part to allow it to redo the action.

None.

-----

## RedoAction - Remarks

A part may need to give the user the capability of repeating the effects of recently undone commands. OpenDoc calls this method when the user chooses to redo an action of this part.

Your part is responsible for notifying the clipboard whenever a cut, copy, or paste operation is done, undone, or redone. If your part's RedoAction method is called, it should remove the object from its content model and notify the clipboard that the cut was redone.

-----

## RedoAction - Exception Handling

kODErrDoesNotUndo

The specified action is not an undo action of this part.

-----

## RedoAction - Override Policy

If you subclass [ODPart](#), you can override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## RedoAction - Related Methods

### Related Methods

- [ODPart::UndoAction](#)
- 

## RedoAction - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## ReleaseRealPart

---

## ReleaseRealPart - Syntax

This method releases the part object encapsulated by the part wrapper.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>
```

```
ReleaseRealPart ();
```

---

## ReleaseRealPart - Return Value

None.

---

## ReleaseRealPart - Parameters

None.

---

## ReleaseRealPart - Remarks

For part wrappers, this method marks the part object as available for access by another client. Your part's ReleaseRealPart method is almost never called unless you really need to access the actual part.

---

## ReleaseRealPart - Exception Handling

kODErrPartNotWrapper

This method was called on an actual part, not a part wrapper.

---

## ReleaseRealPart - Override Policy

If you subclass [ODPart](#), you must not override this method.

---

## ReleaseRealPart - Related Methods

### Related Methods

- [ODPart::GetRealPart](#)
  - [ODPart::IsRealPart](#)
- 

## ReleaseRealPart - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

---

# RemoveEmbeddedFrame

---

## RemoveEmbeddedFrame - Syntax

This method removes the specified embedded frame from this part's content.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODFrame      *embeddedFrame;

RemoveEmbeddedFrame (embeddedFrame);
```

---

## RemoveEmbeddedFrame Parameter - embeddedFrame

**embeddedFrame** (ODFrame \*) - input  
A reference to a embedded frame of this part.

---

## RemoveEmbeddedFrame - Return Value

None.

---

## RemoveEmbeddedFrame - Parameters

**embeddedFrame** (ODFrame \*) - input  
A reference to a embedded frame of this part.

None.

---

## RemoveEmbeddedFrame - Remarks



An embedded part calls this part's `RemoveEmbeddedFrame` method when it no longer wants to display itself in the specified frame.

If your part supports embedding, your part's `RemoveEmbeddedFrame` method should respond to a request to remove an embedded frame. Your part's `RemoveEmbeddedFrame` method should remove only frames that were added by calls to your part's [RequestEmbeddedFrame](#) method.

---

## RemoveEmbeddedFrame - Exception Handling

`kODErrCannotEmbed`

This part does not support embedding.

`kODErrInvalidFrame`

The specified frame is not an embedded frame of this part.

---

## RemoveEmbeddedFrame - Override Policy

If you subclass [ODPart](#), you can override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely. This method needs to be implemented only by container parts.

---

## RemoveEmbeddedFrame - Related Methods

### Related Methods

- [ODPart::RequestEmbeddedFrame](#)

---

## RemoveEmbeddedFrame - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## RequestEmbeddedFrame

---

# RequestEmbeddedFrame - Syntax

This method should create a new display frame for the specified embedded part.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODFrame      *containingFrame;
ODFrame      *baseFrame;
ODShape      *frameShape;
ODPart       *embedPart;
ODTypeToken   viewType;
ODTypeToken   presentation;
ODBoolean     isOverlaid;
ODFrame      *rc;

rc = RequestEmbeddedFrame(containingFrame,
    baseFrame, frameShape, embedPart, viewType,
    presentation, isOverlaid);
```

---

## RequestEmbeddedFrame Parameter - containingFrame

**containingFrame** (ODFrame \*) - input  
A reference to a display frame in which to embed the new frame.

---

## RequestEmbeddedFrame Parameter - baseFrame

**baseFrame** (ODFrame \*) - input  
A reference to a sibling frame of the frame to be created, and also a display frame for the embedded part.

---

## RequestEmbeddedFrame Parameter - frameShape

**frameShape** (ODShape \*) - input  
A reference to a requested shape for the new frame, expressed in frame coordinates of the containing frame.

---

## RequestEmbeddedFrame Parameter - embedPart

**embedPart** (ODPart \*) - input  
A reference to a part that is to be displayed in the new frame.

---

# RequestEmbeddedFrame Parameter - viewType

**viewType** (ODTypeToken) - input

A tokenized string representing the view type to be assigned to this part's frame.

This parameter must be the tokenized form of one of the following view-type constants. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODViewAsFrame	Framed view type.
kODViewAsLargeIcon	Large-icon (standard) view type
kODViewAsSmallIcon	Small-icon view type.
kODViewAsThumbNail	Thumbnail view type

---

# RequestEmbeddedFrame Parameter - presentation

**presentation** (ODTypeToken) - input

A tokenized string representing the presentation to be assigned to this part's frame.

---

# RequestEmbeddedFrame Parameter - isOverlaid

**isOverlaid** (ODBoolean) - input

A flag indicating whether this part's frame is to be an overlaid frame.

kODTrue	The part's frame is to be an overlaid frame.
kODFalse	The part's frame is not to be an overlaid frame.

---

# RequestEmbeddedFrame Return Value - rc

**rc** (ODFrame \*) - returns

A reference to a new frame object.

---

# RequestEmbeddedFrame - Parameters

**containingFrame** (ODFrame \*) - input

A reference to a display frame in which to embed the new frame.

**baseFrame** (ODFrame \*) - input

A reference to a sibling frame of the frame to be created, and also a display frame for the embedded part.

**frameShape** (ODShape \*) - input

A reference to a requested shape for the new frame, expressed in frame coordinates of the containing frame.

**embedPart** (ODPart \*) - input

A reference to a part that is to be displayed in the new frame.

**viewType** (ODTypeToken) - input

A tokenized string representing the view type to be assigned to this part's frame.

This parameter must be the tokenized form of one of the following view-type constants. You can call the session object's [Tokenize](#) method to obtain a token corresponding to one of these constants.

kODViewAsFrame

Framed view type.

kODViewAsLargeIcon

Large-icon (standard) view type

kODViewAsSmallIcon

Small-icon view type.

kODViewAsThumbNail

Thumbnail view type

**presentation** (ODTypeToken) - input

A tokenized string representing the presentation to be assigned to this part's frame.

**isOverlaid** (ODBoolean) - input

A flag indicating whether this part's frame is to be an overlaid frame.

kODTrue

The part's frame is to be an overlaid frame.

kODFalse

The part's frame is not to be an overlaid frame.

**rc** (ODFrame \*) - returns

A reference to a new frame object.

-----

## RequestEmbeddedFrame - Remarks

An embedded part calls its containing part's `RequestEmbeddedFrame` method when it needs an extra frame to flow content into, for example, an additional column or additional page of text. The part must specify one of its current display frames as a base frame; the new frame is a sibling of the base frame, and it is in the same group as the base frame.

If your part supports embedding, it may need to respond to a request to add an embedded frame. Your part's `RequestEmbeddedFrame` method should call your draft's [CreateFrame](#) method to create the new frame and then return it to the embedded part.

Before returning the frame object, your part's `RequestEmbeddedFrame` method should call the frame object's [Acquire](#) method. When the caller has finished using the returned frame object, it should call the frame object's [Release](#) method.

For more information on frame negotiation, see the layout and embedding chapter in the *OpenDoc Programming Guide* .

-----

## RequestEmbeddedFrame - Exception Handling

kODErrCannotEmbed

kODErrInvalidFrame

kODErrOutOfMemory

This part does not support embedding.

The specified sibling frame is not an embedded frame of this part.

There is not enough memory to embed a part.

-----

## RequestEmbeddedFrame - Override Policy

If you subclass [ODPart](#), you can override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely. This method needs to be implemented only by container parts.

---

## RequestEmbeddedFrame - Related Methods

### Related Methods

- [ODDraft::CreateFrame](#)
  - [ODSession::Tokenize](#)
- 

## RequestEmbeddedFrame - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## RequestFrameShape

---

## RequestFrameShape - Syntax

This method negotiates a new frame shape for the specified frame embedded in this part.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODFrame      *embeddedFrame;
ODShape      *frameShape;
ODShape      *rc;

rc = RequestFrameShape(embeddedFrame, frameShape);
```

---

## RequestFrameShape Parameter - embeddedFrame

**embeddedFrame** (ODFrame \*) - input

A reference to an embedded frame for which the frame-shape change is requested.

---

## RequestFrameShape Parameter - frameShape

**frameShape** (ODShape \*) - input

A reference to a requested shape, expressed in the frame coordinates of the embedded frame.

---

## RequestFrameShape Return Value - rc

**rc** (ODShape \*) - returns

A reference to a new shape for the embedded frame, expressed in frame coordinates.

---

## RequestFrameShape - Parameters

**embeddedFrame** (ODFrame \*) - input

A reference to an embedded frame for which the frame-shape change is requested.

**frameShape** (ODShape \*) - input

A reference to a requested shape, expressed in the frame coordinates of the embedded frame.

**rc** (ODShape \*) - returns

A reference to a new shape for the embedded frame, expressed in frame coordinates.

---

## RequestFrameShape - Remarks

OpenDoc calls this method to initiate a frame negotiation process requested by this part's embedded part.

Your part's RequestFrameShape method should decide what new shape to give an embedded frame, using the requested frame shape as a guideline, and return a reference to the shape object it allows the embedded frame to have. The embedded frame stores the shape as its new frame shape and returns the shape to its part so that its part knows what its new shape is. The embedded part must accept the returned shape, although it may make further requests for different shapes or additional frames.

Before returning the shape object, your part's RequestFrameShape method should call the shape object's [Acquire](#) method. When the caller has finished using the returned shape object, it should call the shape object's [Release](#) method.

For more information on frame negotiation, see chapter on the layout and embedding in the *OpenDoc Programming Guide* .

---

## RequestFrameShape - Exception Handling

## RequestFrameShape - Override Policy

If you subclass [ODPart](#), you can override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely. This method needs to be implemented only by container parts.

---

## RequestFrameShape - Related Methods

### Related Methods

- [ODFrame::RequestFrameShape](#)
- 

## RequestFrameShape - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

[Related Methods](#)

---

## RevealFrame

---

## RevealFrame - Syntax

This method makes the specified embedded frame visible by scrolling it into view.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODFrame      *embeddedFrame;
ODShape      *revealShape;
ODBoolean    rc;

rc = RevealFrame(embeddedFrame, revealShape);
```

---

## RevealFrame Parameter - embeddedFrame

**embeddedFrame** (ODFrame \*) - input  
A reference to an embedded frame of this part.

---

## RevealFrame Parameter - revealShape

**revealShape** (ODShape \*) - input  
A reference to a shape object, expressed in frame coordinates, that indicates the portion of the frame to be revealed.

---

## RevealFrame Return Value - rc

**rc** (ODBoolean) - returns  
A flag indicating whether this part was able to reveal the frame.

kODTrue	This part was able to reveal the frame.
kODFalse	This part was not able to reveal the frame.

---

## RevealFrame - Parameters

**embeddedFrame** (ODFrame \*) - input  
A reference to an embedded frame of this part.

**revealShape** (ODShape \*) - input  
A reference to a shape object, expressed in frame coordinates, that indicates the portion of the frame to be revealed.

**rc** (ODBoolean) - returns  
A flag indicating whether this part was able to reveal the frame.

kODTrue	This part was able to reveal the frame.
kODFalse	This part was not able to reveal the frame.

---

## RevealFrame - Remarks

An embedded part calls this part's RevealFrame method when it needs to become visible, such as in conjunction with a keyboard event which requires an undisplayed area of a window to scroll into view.

Your part's RevealFrame method should scroll one of your part's display frames, if necessary, to make the specified embedded frame visible.



If your part has no visible display frames, it should ask its containing part to reveal one of them. If your part has no display frames, or if your part's containing frame cannot reveal the display frame, your method should open a frame in a new window.

---

## RevealFrame - Exception Handling

kODErrCannotEmbed

This part does not support embedding.

kODErrInvalidFrame

The specified frame is not an embedded frame of this part.

---

## RevealFrame - Override Policy

If you subclass [ODPart](#), you can override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely. This method needs to be implemented only by container parts.

---

## RevealFrame - Related Methods

### Related Methods

- [ODFrame::AcquireFrameShape](#)
- 

## RevealFrame - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

[Related Methods](#)

---

## RevealLink

---

## RevealLink - Syntax

This method shows the content at the source of a link.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODLinkSource      *linkSource;

RevealLink(linkSource);
```

-----

## RevealLink Parameter - linkSource

**linkSource** (ODLinkSource \*) - input  
A reference to a link-source object representing the linked content to be revealed.

-----

## RevealLink - Return Value

None.

-----

## RevealLink - Parameters

**linkSource** (ODLinkSource \*) - input  
A reference to a link-source object representing the linked content to be revealed.

None.

-----

## RevealLink - Remarks

OpenDoc calls this method when a link source needs to be shown; this method is not called by parts.

Your part's `RevealLink` method should select the linked content to be revealed in one of its frames, scroll the frame into view using its containing part's `RevealFrame` method, and make that frame active. If your part has no visible display frames, or if your part's containing frame cannot reveal the display frame, your method should open a frame in a new window.

-----

## RevealLink - Exception Handling

<code>kODErrDoesNotLink</code>	The specified link is not a link of this part.
--------------------------------	--

-----

# RevealLink - Override Policy

If you subclass [ODPart](#), you can override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## RevealLink - Related Methods

### Related Methods

- [ODPart::RevealFrame](#)
- 

## RevealLink - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## SequenceChanged

---

## SequenceChanged - Syntax

This method should notify this part that the sequencing of this part's display frame within its frame group has changed.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODFrame      *frame;

SequenceChanged (frame);
```

---

## SequenceChanged Parameter - frame

**frame** (ODFrame \*) - input  
A reference to a display frame whose sequence has been reordered.

---

## SequenceChanged - Return Value

None.

---

## SequenceChanged - Parameters

**frame** (ODFrame \*) - input  
A reference to a display frame whose sequence has been reordered.

None.

---

## SequenceChanged - Remarks

OpenDoc calls this method when this part's containing part adds a new frame to the group or reorders the frames in the group. The containing part calls its embedded frame's [ChangeSequenceNumber](#) method to initiate the change.

---

## SequenceChanged - Exception Handling

kODErrInvalidFrame	The specified frame is not a display frame of this part.
--------------------	--

---

## SequenceChanged - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## SequenceChanged - Related Methods

### Related Methods

- [ODFrame::ChangeSequenceNumber](#)

---

## SequenceChanged - Topics

### Class:

ODPart

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

[Related Methods](#)

---

## ShowLink (OS/2)

---

### ShowLink (OS/2) - Syntax

This method is used by the *Link* page of the Properties notebook to indicate that the user requests the part to show the link.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>
```

```
ODLink      *link;
```

```
ShowLink(link);
```

---

### ShowLink (OS/2) Parameter - link

**link** (ODLink \*) - input

The link object to be shown.

---

### ShowLink (OS/2) - Return Value

None.

---

## ShowLink (OS/2) - Parameters

**link** (ODLink \*) - input  
The link object to be shown.

None.

---

## ShowLink (OS/2) - Remarks

There is no default action in this method except to raise an exception kODErrSubClassResponsibility.

---

## ShowLink (OS/2) - Exception Handling

kODErrSubClassResponsibility

The subclass must override this method.

---

## ShowLink (OS/2) - Override Policy

If you subclass [ODPart](#), you must override this method.

---

## ShowLink (OS/2) - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

---

## UndoAction

---

## UndoAction - Syntax

This method undoes the specified action.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODActionData      *actionState;

UndoAction(actionState);
```

-----

## UndoAction Parameter - actionState

**actionState** (ODActionData \*) - input

A byte array whose buffer contains the data previously logged by this part to allow it to undo the action.

-----

## UndoAction - Return Value

None.

-----

## UndoAction - Parameters

**actionState** (ODActionData \*) - input

A byte array whose buffer contains the data previously logged by this part to allow it to undo the action.

None.

-----

## UndoAction - Remarks

OpenDoc calls this method when the user chooses to undo an action of this part.

Your part may need to give the user the capability of reversing the effects of recently executed commands. If it does, your part's UndoAction method should perform any reverse editing necessary to restore itself to the state it possessed before the specified action.

Your part is responsible for notifying the clipboard whenever a cut, copy, or paste operation is done, undone, or redone. When a part cuts an object to the clipboard, a reference to the object should be saved in an undo action. If your part's UndoAction method is called, it should reinstate the object into its content model from its undo information and notify the clipboard that the cut was undone.

-----

## UndoAction - Exception Handling

## UndoAction - Override Policy

If you subclass [ODPart](#), you can override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## UndoAction - Related Methods

### Related Methods

- [ODPart::RedoAction](#)
- 

## UndoAction - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

[Related Methods](#)

---

## UpdateFromLinkSource (OS/2)

---

## UpdateFromLinkSource (OS/2) - Syntax

This method is used by the *Link Source* page of the Properties notebook to indicate that the user requested the part to update the link source. As a side-effect, the registered links also get updated.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>
```

```
ODLinkSource    *linkSource;
```

```
UpdateFromLinkSource(linkSource);
```



---

## UpdateFromLinkSource (OS/2) Parameter - linkSource

**linkSource** (ODLinkSource \*) - input

The link source object whose links are to be updated.

---

## UpdateFromLinkSource (OS/2) - Return Value

None.

---

## UpdateFromLinkSource (OS/2) - Parameters

**linkSource** (ODLinkSource \*) - input

The link source object whose links are to be updated.

None.

---

## UpdateFromLinkSource (OS/2) - Remarks

There is no default action in this method except to raise an exception kODErrSubClassResponsibility.

---

## UpdateFromLinkSource (OS/2) - Exception Handling

kODErrSubClassResponsibility

The subclass must override this method.

---

## UpdateFromLinkSource (OS/2) - Override Policy

If you subclass [ODPart](#), you must override this method.

---

## UpdateFromLinkSource (OS/2) - Topics

**Class:**

ODPart

Select an item:

[Syntax](#)[Parameters](#)[Returns](#)[Remarks](#)[Override Policy](#)[Exception Handling](#)

---

## UsedShapeChanged

---

### UsedShapeChanged - Syntax

This method notifies this part that the used shape of one of its embedded frames has changed.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODFrame      *embeddedFrame;

UsedShapeChanged (embeddedFrame);
```

---

### UsedShapeChanged Parameter - embeddedFrame

**embeddedFrame** (ODFrame \*) - input

A reference to an embedded frame of this part.

---

### UsedShapeChanged - Return Value

None.

---

### UsedShapeChanged - Parameters

**embeddedFrame** (ODFrame \*) - input

A reference to an embedded frame of this part.

None.

-----

## UsedShapeChanged - Remarks

OpenDoc calls this method when a frame embedded in this part changes its used shape. Containing parts that have wrapped their content to the contour of an embedded frame's used shape (as in text wrapping) need to adjust the layout of that content for the new used shape.

-----

## UsedShapeChanged - Exception Handling

kODErrCannotEmbed

This part does not support embedding.

kODErrInvalidFrame

The specified frame is not an embedded frame of this part.

-----

## UsedShapeChanged - Override Policy

If you subclass [ODPart](#), you can override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely. This method needs to be implemented only by container parts.

-----

## UsedShapeChanged - Related Methods

### Related Methods

- [ODFrame::ChangeUsedShape](#)

-----

## UsedShapeChanged - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

[Exception Handling](#)

[Related Methods](#)

-----

## ViewTypeChanged

---

## ViewTypeChanged - Syntax

This method should notify this part that the view type of one of its display frames has changed.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>

ODFrame      *frame;

ViewTypeChanged(frame);
```

---

## ViewTypeChanged Parameter - frame

**frame** (ODFrame \*) - input  
A reference to a display frame for this part.

---

## ViewTypeChanged - Return Value

None.

---

## ViewTypeChanged - Parameters

**frame** (ODFrame \*) - input  
A reference to a display frame for this part.

None.

---

## ViewTypeChanged - Remarks

OpenDoc calls this method when this part's display frame changes its view type.

Your part's ViewTypeChanged method should examine the new view type using its frame's [GetViewType](#) method. It should then display itself in the specified display frame according to the indicated view type. If your part does not support the requested view type, it should instead pick a view type that it can support and then call its frame's [SetViewType](#) method to update the view type in the frame.

---

## ViewTypeChanged - Exception Handling

kODErrInvalidFrame

The specified frame is not a display frame of this part.

---

## ViewTypeChanged - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## ViewTypeChanged - Related Methods

### Related Methods

- [ODFrame::GetViewType](#)
  - [ODFrame::SetViewType](#)
- 

## ViewTypeChanged - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Exception Handling](#)  
[Related Methods](#)

---

## WriteActionState

---

## WriteActionState - Syntax

This method writes the undo action data into the specified storage-unit view object.

```
#define INCL_ODPART
```

```
#define INCL_ODAPI
#include <os2.h>

ODActionData      *actionState;
ODStorageUnitView *storageUnitView;

WriteActionState(actionState, storageUnitView);
```

-----

## WriteActionState Parameter - actionState

**actionState** (ODActionData \*) - input

A byte array whose buffer contains the data previously logged by this part to allow it to redo the action.

-----

## WriteActionState Parameter - storageUnitView

**storageUnitView** (ODStorageUnitView \*) - input

A reference to a storage-unit view object where the action data is to be written to storage.

-----

## WriteActionState - Return Value

None.

-----

## WriteActionState - Parameters

**actionState** (ODActionData \*) - input

A byte array whose buffer contains the data previously logged by this part to allow it to redo the action.

**storageUnitView** (ODStorageUnitView \*) - input

A reference to a storage-unit view object where the action data is to be written to storage.

None.

-----

## WriteActionState - Remarks

OpenDoc calls this method when it writes the undo action data to persistent storage. Your part's WriteActionState method will not be called in OpenDoc 1.0 due to the lack of a persistent undo model; it is provided here for compatibility with future versions of OpenDoc.

-----

# WriteActionState - Exception Handling

kODErrDoesNotUndo

The specified action is not an undo action of this part.

---

# WriteActionState - Override Policy

If you subclass [ODPart](#), you can override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

# WriteActionState - Related Methods

## Related Methods

- [ODPart::DisposeActionState](#)
  - [ODPart::ReadActionState](#)
- 

# WriteActionState - Topics

## Class:

[ODPart](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Exception Handling](#)  
[Related Methods](#)

---

# WritePartInfo

---

# WritePartInfo - Syntax

This method should write the part-information data for a display frame of this part into the specified storage-unit view object.

```
#define INCL_ODPART
#define INCL_ODAPI
#include <os2.h>
```

```
ODInfoType      partInfo;  
ODStorageUnitView *storageUnitView;  
  
WritePartInfo(partInfo, storageUnitView);
```

---

## WritePartInfo Parameter - partInfo

**partInfo** ([ODInfoType](#)) - input  
The part-information data to write.

---

## WritePartInfo Parameter - storageUnitView

**storageUnitView** ([ODStorageUnitView \\*](#)) - input  
A reference to a storage-unit view object that is focused to the frame's part-information property, but not to any value within that property.

---

## WritePartInfo - Return Value

None.

---

## WritePartInfo - Parameters

**partInfo** ([ODInfoType](#)) - input  
The part-information data to write.

**storageUnitView** ([ODStorageUnitView \\*](#)) - input  
A reference to a storage-unit view object that is focused to the frame's part-information property, but not to any value within that property.

None.

---

## WritePartInfo - Remarks

When a document is saved (using the **Save** command from the **Document** menu), OpenDoc may ask this part to save a display frame's part-information data to a particular storage unit. OpenDoc calls this method when it writes out the part-information data for a display frame of a part.



Your part's WritePartInfo method should get the storage unit associated with the specified storage-unit view and focus it to the values in the frame's part-information property as necessary to write the formats it needs.

---

## WritePartInfo - Override Policy

If you subclass [ODPart](#), you must override this method. Your override method must not call its inherited method; that is, your override method must implement this method's functionality completely.

---

## WritePartInfo - Related Methods

### Related Methods

- [ODPart::ClonePartInfo](#)
  - [ODPart::ReadPartInfo](#)
- 

## WritePartInfo - Topics

### Class:

[ODPart](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Related Methods](#)

---

## ODPartHandlerInfo

**Class Definition File:** PARTINFO.IDL

### Class Hierarchy

SOMObject  
    **ODPartHandlerInfo**

### Description

An object of the ODPartHandlerInfo class gathers and provides all of the information necessary for OpenDoc to perform dynamic binding.

An ODPartHandlerInfo object is created for each registered part handler. When a new part handler is being registered, the ODPartHandlerInfo object creates an instance of the part's metaclass (inherited from [M\\_ODPart](#)), and calls its methods. The object can then be queried for the part handler's class name, handler name, display name, DLL file name, OLE2 class ID, and information for all of the part kinds which the handler supports.

### Methods

The methods defined by the ODPartHandlerInfo class include:

- [GetDLLName](#)
- [GetOLE2ClassId](#)
- [GetOperatingSystem](#)

- [GetPartHandlerClassName](#)
- [GetPartHandlerDisplayName](#)
- [GetPartHandlerName](#)
- [GetPartKindInfo](#)
- [GetWindowsIconFileName](#)
- [SetOperatingSystem](#)

## Overridden Methods

There are currently no methods overridden by the ODPartHandlerInfo class.

---

## GetDLLName (OS/2)

---

## GetDLLName (OS/2) - Syntax

This method returns the DLL name for this part handler.

```
#define INCL_ODPARTHANDLERINFO
#define INCL_ODAPI
#include <os2.h>

string    rv;

rv = GetDLLName();
```

---

## GetDLLName (OS/2) Return Value - rv

**rv** ([string](#)) - returns

The DLL name for this part handler. When it is finished using this value, the client code is responsible for freeing it by calling the somFree method.

---

## GetDLLName (OS/2) - Parameters

**rv** ([string](#)) - returns

The DLL name for this part handler. When it is finished using this value, the client code is responsible for freeing it by calling the somFree method.

---

## GetDLLName (OS/2) - Topics

### Class:

ODPartHandlerInfo

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

---

## GetOLE2ClassId (OS/2)

---

### GetOLE2ClassId (OS/2) - Syntax

This method returns the OLE2 class ID.

```
#define INCL_ODPARTHANDLERINFO
#define INCL_ODAPI
#include <os2.h>

string    rv;

rv = GetOLE2ClassId();
```

---

### GetOLE2ClassId (OS/2) Return Value - rv

**rv** ([string](#)) - returns

The OLE2 class ID. When it is finished using this value, the client code is responsible for freeing it by calling SOMFree.

---

### GetOLE2ClassId (OS/2) - Parameters

**rv** ([string](#)) - returns

The OLE2 class ID. When it is finished using this value, the client code is responsible for freeing it by calling SOMFree.

---

### GetOLE2ClassId (OS/2) - Remarks

This method is supplied for interoperability with OLE2 running in a WIN-OS/2 environment.

---

### GetOLE2ClassId (OS/2) - Topics

**Class:**

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

---

## GetOperatingSystem (OS/2)

---

### GetOperatingSystem (OS/2) - Syntax

This method returns the operating system on which this part handler runs.

```
#define INCL_ODPARTHANDLERINFO
#define INCL_ODAPI
#include <os2.h>
```

```
OperatingSystem    rv;
```

```
rv = GetOperatingSystem();
```

---

### GetOperatingSystem (OS/2) Return Value - rv

**rv** ([OperatingSystem](#)) - returns

A code indicating operating system on which this part runs. This parameter can be set to one of the following values:

AIX	Advanced Interaction Executive
MAC	MacIntosh
OS2	Operating System/2
WINDOWS	Microsoft Windows

---

### GetOperatingSystem (OS/2) - Parameters

**rv** ([OperatingSystem](#)) - returns

A code indicating operating system on which this part runs. This parameter can be set to one of the following values:

AIX	Advanced Interaction Executive
MAC	MacIntosh
OS2	Operating System/2

## GetOperatingSystem (OS/2) - Topics

**Class:**

ODPartHandlerInfo

Select an item:

[Syntax](#)[Parameters](#)[Returns](#)

---

## GetPartHandlerClassName (OS/2)

---

## GetPartHandlerClassName (OS/2) - Syntax

This method returns the SOM class name for this part handler.

```
#define INCL_ODPARTHANDLERINFO
#define INCL_ODAPI
#include <os2.h>

string    rv;

rv = GetPartHandlerClassName();
```

---

## GetPartHandlerClassName (OS/2) Return Value - rv

**rv** ([string](#)) - returns

The SOM class for this part handler. When it is finished using this value, the client code is responsible for freeing it by calling somFree.

---

## GetPartHandlerClassName (OS/2) - Parameters

**rv** ([string](#)) - returns

The SOM class for this part handler. When it is finished using this value, the client code is responsible for freeing it by calling somFree.

---

## GetPartHandlerClassName (OS/2) - Topics

**Class:**  
ODPartHandlerInfo

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## GetPartHandlerDisplayName (OS/2)

---

### GetPartHandlerDisplayName (OS/2) - Syntax

This method returns the display name of the part handler.

```
#define INCL_ODPARTHANDLERINFO
#define INCL_ODAPI
#include <os2.h>

string    rv;

rv = GetPartHandlerDisplayName();
```

---

### GetPartHandlerDisplayName (OS/2) Return Value - rv

**rv** ([string](#)) - returns

The name to be displayed for the part handler.

For example, "IBM Movie Editor".

When it is finished using this value, the client code is responsible for freeing it by calling SOMFree.

---

### GetPartHandlerDisplayName (OS/2) - Parameters

**rv** ([string](#)) - returns

The name to be displayed for the part handler.

For example, "IBM Movie Editor".

When it is finished using this value, the client code is responsible for freeing it by calling SOMFree.

---

### GetPartHandlerDisplayName (OS/2) - Topics

**Class:**  
ODPartHandlerInfo

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## GetPartHandlerName (OS/2)

---

### GetPartHandlerName (OS/2) - Syntax

This method returns the name of the part handler.

```
#define INCL_ODPARTHANDLERINFO
#define INCL_ODAPI
#include <os2.h>

ISOString    rv;

rv = GetPartHandlerName();
```

---

### GetPartHandlerName (OS/2) Return Value - rv

**rv** ([ISOString](#)) - returns  
A string containing the name of the part handler.

For example, `IBM:Viewer:Sample:Movie`

When it is finished using this value, the client code is responsible for freeing it by calling `SOMFree`.

---

### GetPartHandlerName (OS/2) - Parameters

**rv** ([ISOString](#)) - returns  
A string containing the name of the part handler.

For example, `IBM:Viewer:Sample:Movie`

When it is finished using this value, the client code is responsible for freeing it by calling `SOMFree`.

---

### GetPartHandlerName (OS/2) - Topics

**Class:**  
ODPartHandlerInfo

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## GetPartKindInfo (OS/2)

---

### GetPartKindInfo (OS/2) - Syntax

This method returns the part kinds that are supported by the part handler and meets the specified search criteria.

```
#define INCL_ODPARTHANDLERINFO
#define INCL_ODAPI
#include <os2.h>

string          category;
_IDL_SEQUENCE_PartKindInfo rv;

rv = GetPartKindInfo(category);
```

---

### GetPartKindInfo (OS/2) Parameter - category

**category** ([string](#)) - input  
The category to be used as search criteria.

---

### GetPartKindInfo (OS/2) Return Value - rv

**rv** ([\\_IDL\\_SEQUENCE\\_PartKindInfo](#)) - returns  
The part kinds supported by this part handler.

If *category* is supplied, only those part kinds that belong to this category are returned. If *category* is NULL, then all part kinds for this part editor are returned.

When it is finished using this sequence, the client code is responsible for freeing each object in the sequence by calling SOMFree and freeing the sequence buffer.

---

### GetPartKindInfo (OS/2) - Parameters



**category** ([string](#)) - input  
The category to be used as search criteria.

**rv** ([\\_IDL\\_SEQUENCE\\_PartKindInfo](#)) - returns  
The part kinds supported by this part handler.

If *category* is supplied, only those part kinds that belong to this category are returned. If *category* is NULL, then all part kinds for this part editor are returned.

When it is finished using this sequence, the client code is responsible for freeing each object in the sequence by calling SOMFree and freeing the sequence buffer.

-----

## GetPartKindInfo (OS/2) - Topics

**Class:**  
ODPartHandlerInfo

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)

-----

## GetWindowsIconFileName (OS/2)

-----

## GetWindowsIconFileName (OS/2) - Syntax

This method returns the name of the Windows icon file.

```
#define INCL_ODPARTHANDLERINFO
#define INCL_ODAPI
#include <os2.h>

string    rv;

rv = GetWindowsIconFileName();
```

-----

## GetWindowsIconFileName (OS/2) Return Value - rv

**rv** ([string](#)) - returns  
The name of the Windows icon file. When it is finished using this value, the client code is responsible for freeing it by calling SOMFree.

-----

## GetWindowsIconFileName (OS/2) - Parameters

**rv** ([string](#)) - returns

The name of the Windows icon file. When it is finished using this value, the client code is responsible for freeing it by calling SOMFree.

---

## GetWindowsIconFileName (OS/2) - Remarks

This method is supplied for interoperability with OLE2 in a WIN-OS/2 environment.

---

## GetWindowsIconFileName (OS/2) - Topics

### Class:

ODPartHandlerInfo

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

---

## SetOperatingSystem (OS/2)

---

## SetOperatingSystem (OS/2) - Syntax

This method sets the operating system that this part runs on.

```
#define INCL_ODPARTHANDLERINFO
#define INCL_ODAPI
#include <os2.h>
```

```
OperatingSystem    os;
```

```
SetOperatingSystem(os);
```

---

## SetOperatingSystem (OS/2) Parameter - os

**os** ([OperatingSystem](#)) - input

A code indicating on which operating system this part is to run. This parameter can be set to one of the following values:

AIX

Advanced Interaction Executive

MAC

OS2	MacIntosh
WINDOWS	Operating System/2
	Microsoft Windows

-----

## SetOperatingSystem (OS/2) - Return Value

None.

-----

## SetOperatingSystem (OS/2) - Parameters

**os** ([OperatingSystem](#)) - input

A code indicating on which operating system this part is to run. This parameter can be set to one of the following values:

AIX	Advanced Interaction Executive
MAC	MacIntosh
OS2	Operating System/2
WINDOWS	Microsoft Windows

None.

-----

## SetOperatingSystem (OS/2) - Topics

**Class:**

ODPartHandlerInfo

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)

-----

## ODPartHandlerRegistry

**Class Definition File:** ODPRTREG.IDL

**Class Hierarchy**

SOMObject  
**ODPartHandlerRegistry**

**Description**

This class is the main interface for part registration. Its methods allow you to query and update the registration information.

## Methods

The methods defined by the ODPartHandlerRegistry class include:

- [DeregisterPartHandler](#)
- [DeregisterPartHandlerClass](#)
- [GetCategoryList](#)
- [GetFileExtensionList](#)
- [GetFileTypeList](#)
- [GetPartHandlerInfo](#)
- [GetPartHandlerList](#)
- [GetPartHandlerListForFileExt](#)
- [GetPartHandlerListForFileTypes](#)
- [GetPartKindList](#)
- [GetPreferredPartHandler](#)
- [GetPreferredPartHandlerForCategory](#)
- [GetPreferredPartHandlerForFileExt](#)
- [GetPreferredPartHandlerForFileType](#)
- [RegisterPartHandlerClass](#)
- [SetPreferredPartHandler](#)
- [SetPreferredPartHandlerForCategory](#)
- [SetPreferredPartHandlerForFileExt](#)
- [SetPreferredPartHandlerForFileType](#)

## Overridden Methods

There are currently no methods overridden by the ODPartHandlerRegistry class.

---

## DeregisterPartHandler (OS/2)

---

## DeregisterPartHandler (OS/2) - Syntax

This method deregisters a part handler with the OpenDoc registry.

```
#define INCL_ODPARTHANDLERREGISTRY
#define INCL_ODAPI
#include <os2.h>

ISOString    partHandlerName;
long         rv;          /* Return codes. */

rv = DeregisterPartHandler(partHandlerName);
```

---

## DeregisterPartHandler (OS/2) Parameter - partHandlerName

**partHandlerName** ([ISOString](#)) - input

The name of the part handler to be deregistered.

---

## DeregisterPartHandler (OS/2) Return Value - rv

**rv** (long) - returns  
Return codes.

NO\_ERROR

ERROR\_COULD\_NOT\_FIND\_PART\_HANDLER

ERROR\_COUNND\_NOT\_FIND\_CLASS

-----

## DeregisterPartHandler (OS/2) - Parameters

**partHandlerName** ([ISOString](#)) - input  
The name of the part handler to be deregistered.

**rv** (long) - returns  
Return codes.

NO\_ERROR

ERROR\_COULD\_NOT\_FIND\_PART\_HANDLER

ERROR\_COUNND\_NOT\_FIND\_CLASS

-----

## DeregisterPartHandler (OS/2) - Topics

**Class:**  
ODPartHandlerRegistry

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

-----

## DeregisterPartHandlerClass (OS/2)

-----

## DeregisterPartHandlerClass (OS/2) - Syntax

This method deregisters a part handler class with the OpenDoc registry.

```
#define INCL_ODPARTHANDLERREGISTRY
#define INCL_ODAPI
#include <os2.h>

string    partHandlerClassName;
```

```
long         rv;                                /* Return codes. */  
  
rv = DeregisterPartHandlerClass(partHandlerClassName);
```

-----

## DeregisterPartHandlerClass (OS/2) Parameter - partHandlerClassN

**partHandlerClassName** ([string](#)) - input  
The class name of the part handler to be deregistered.

-----

## DeregisterPartHandlerClass (OS/2) Return Value - rv

**rv** (long) - returns  
Return codes.

- NO\_ERROR
- ERROR\_COULD\_NOT\_FIND\_PART\_HANDLER
- ERROR\_COUNED\_NOT\_FIND\_CLASS

-----

## DeregisterPartHandlerClass (OS/2) - Parameters

**partHandlerClassName** ([string](#)) - input  
The class name of the part handler to be deregistered.

**rv** (long) - returns  
Return codes.

- NO\_ERROR
- ERROR\_COULD\_NOT\_FIND\_PART\_HANDLER
- ERROR\_COUNED\_NOT\_FIND\_CLASS

-----

## DeregisterPartHandlerClass (OS/2) - Topics

**Class:**  
ODPartHandlerRegistry

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)

-----

# GetCategoryList (OS/2)

---

## GetCategoryList (OS/2) - Syntax

This method returns a list of categories for the specified part handler and kind.

```
#define INCL_ODPARTHANDLERREGISTRY
#define INCL_ODAPI
#include <os2.h>

ISOSString      partHandlerName;
ISOSString      partKindName;
_IDL_SEQUENCE_string rv;

rv = GetCategoryList(partHandlerName, partKindName);
```

---

## GetCategoryList (OS/2) Parameter - partHandlerName

**partHandlerName** ([ISOSString](#)) - input  
The name of the part handler.

---

## GetCategoryList (OS/2) Parameter - partKindName

**partKindName** ([ISOSString](#)) - input  
The name of the part kind.

---

## GetCategoryList (OS/2) Return Value - rv

**rv** ([\\_IDL\\_SEQUENCE\\_string](#)) - returns  
The list of categories.

---

## GetCategoryList (OS/2) - Parameters

**partHandlerName** ([ISOSString](#)) - input  
The name of the part handler.

**partKindName** ([ISOString](#)) - input  
The name of the part kind.

**rv** ([\\_IDL\\_SEQUENCE\\_string](#)) - returns  
The list of categories.

---

## GetCategoryList (OS/2) - Topics

**Class:**  
ODPartHandlerRegistry

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

---

## GetFileExtensionList (OS/2)

---

## GetFileExtensionList (OS/2) - Syntax

This method returns a list of file extensions using the part handler and part kind as search criteria.

```
#define INCL_ODPARTHANDLERREGISTRY
#define INCL_ODAPI
#include <os2.h>

ISOString      partHandlerName;
ISOString      partKindName;
_IDL_SEQUENCE_string rv;

rv = GetFileExtensionList(partHandlerName,
    partKindName);
```

---

## GetFileExtensionList (OS/2) Parameter - partHandlerName

**partHandlerName** ([ISOString](#)) - input  
The part handler which edits the file extension.

---

## GetFileExtensionList (OS/2) Parameter - partKindName



**partKindName** ([ISOString](#)) - input

The part kinds to which the file extensions belong. This parameter is optional and can be set to kODNULL.

---

## GetFileExtensionList (OS/2) Return Value - rv

**rv** ([\\_IDL\\_SEQUENCE\\_string](#)) - returns

A list of file extensions that meet the specified criteria. If the part handler and part kind are not specified, a list of all file extensions for this part handler is returned.

When it is finished using this value, the client code is responsible for freeing each object in the sequence (using SOMFree) and the sequence buffer.

---

## GetFileExtensionList (OS/2) - Parameters

**partHandlerName** ([ISOString](#)) - input

The part handler which edits the file extension.

**partKindName** ([ISOString](#)) - input

The part kinds to which the file extensions belong. This parameter is optional and can be set to kODNULL.

**rv** ([\\_IDL\\_SEQUENCE\\_string](#)) - returns

A list of file extensions that meet the specified criteria. If the part handler and part kind are not specified, a list of all file extensions for this part handler is returned.

When it is finished using this value, the client code is responsible for freeing each object in the sequence (using SOMFree) and the sequence buffer.

---

## GetFileExtensionList (OS/2) - Topics

### Class:

ODPartHandlerRegistry

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

---

## GetFileTypeList (OS/2)

---

## GetFileTypeList (OS/2) - Syntax

This method returns a list of file types for this handler using the specified part handler and part kind as search criteria.

```

#define INCL_ODPARTHANDLERREGISTRY
#define INCL_ODAPI
#include <os2.h>

ISOString      partHandlerName;
ISOString      partKindName;
_IDL_SEQUENCE_string rv;

rv = GetFileTypeList(partHandlerName, partKindName);

```

-----

## GetFileTypeList (OS/2) Parameter - partHandlerName

**partHandlerName** (ISOString) - input  
The part handler which edits the file type.

-----

## GetFileTypeList (OS/2) Parameter - partKindName

**partKindName** (ISOString) - input  
The part kinds to which the file types belong. This parameter is optional and can be set to KODNULL.

-----

## GetFileTypeList (OS/2) Return Value - rv

**rv** (\_IDL\_SEQUENCE\_string) - returns  
A list of file types that meet the specified criteria. If the part handler and part kind are not specified, a list of all file types for this part handler is returned.

When it is finished using this value, the client code is responsible for freeing each object in the sequence (using SOMFree) and the sequence buffer.

-----

## GetFileTypeList (OS/2) - Parameters

**partHandlerName** (ISOString) - input  
The part handler which edits the file type.

**partKindName** (ISOString) - input  
The part kinds to which the file types belong. This parameter is optional and can be set to KODNULL.

**rv** (\_IDL\_SEQUENCE\_string) - returns  
A list of file types that meet the specified criteria. If the part handler and part kind are not specified, a list of all file types for this part handler is returned.

When it is finished using this value, the client code is responsible for freeing each object in the sequence (using SOMFree) and the sequence buffer.

---

## GetFileTypeList (OS/2) - Topics

### Class:

ODPartHandlerRegistry

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

---

## GetPartHandlerInfo (OS/2)

---

### GetPartHandlerInfo (OS/2) - Syntax

This method returns information about the specified part handler.

```
#define INCL_ODPARTHANDLERREGISTRY
#define INCL_ODAPI
#include <os2.h>

ISOString          partHandlerName;
ODPartHandlerInfo  *rv;

rv = GetPartHandlerInfo(partHandlerName);
```

---

### GetPartHandlerInfo (OS/2) Parameter - partHandlerName

**partHandlerName** ([ISOString](#)) - input

The part handler for which information is to be returned.

---

### GetPartHandlerInfo (OS/2) Return Value - rv

**rv** (ODPartHandlerInfo \*) - returns

Information about the specified part handler. When it is finished using this value, the client code is responsible for freeing it by calling SOMFree.

---

### GetPartHandlerInfo (OS/2) - Parameters

**partHandlerName** ([ISOString](#)) - input

The part handler for which information is to be returned.

**rv** (ODPartHandlerInfo \*) - returns

Information about the specified part handler. When it is finished using this value, the client code is responsible for freeing it by calling SOMFree.

-----

## GetPartHandlerInfo (OS/2) - Topics

### Class:

ODPartHandlerRegistry

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

-----

## GetPartHandlerList (OS/2)

-----

## GetPartHandlerList (OS/2) - Syntax

This method returns a list of part-handler names that meet the given criteria.

```
#define INCL_ODPARTHANDLERREGISTRY
#define INCL_ODAPI
#include <os2.h>

ISOString          partKindName;
string             category;
_IDL_SEQUENCE_string rv;

rv = GetPartHandlerList(partKindName, category);
```

-----

## GetPartHandlerList (OS/2) Parameter - partKindName

**partKindName** ([ISOString](#)) - input

The part kind that the part handlers can edit. This parameter is optional and can be set to KODNULL.

-----

## GetPartHandlerList (OS/2) Parameter - category

**category** ([string](#)) - input

A category to which the part kinds that the part handlers can edit belong. This parameter is optional and can be set to kODNULL.

---

## GetPartHandlerList (OS/2) Return Value - rv

**rv** ([\\_IDL\\_SEQUENCE\\_string](#)) - returns

A list of part handlers that meet the search criteria. If the part kind and category are not specified, a list of all part handlers is returned.

The client code is responsible for freeing each object in the sequence (by calling SOMFree) and the sequence buffer.

---

## GetPartHandlerList (OS/2) - Parameters

**partKindName** ([ISOString](#)) - input

The part kind that the part handlers can edit. This parameter is optional and can be set to kODNULL.

**category** ([string](#)) - input

A category to which the part kinds that the part handlers can edit belong. This parameter is optional and can be set to kODNULL.

**rv** ([\\_IDL\\_SEQUENCE\\_string](#)) - returns

A list of part handlers that meet the search criteria. If the part kind and category are not specified, a list of all part handlers is returned.

The client code is responsible for freeing each object in the sequence (by calling SOMFree) and the sequence buffer.

---

## GetPartHandlerList (OS/2) - Topics

### Class:

ODPartHandlerRegistry

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

---

## GetPartHandlerListForFileExt (OS/2)

---

## GetPartHandlerListForFileExt (OS/2) - Syntax

This method returns a list of part-handler names for the specified file extension.

```

#define INCL_ODPARTHANDLERREGISTRY
#define INCL_ODAPI
#include <os2.h>

string          fileExtension;
_IDL_SEQUENCE_string  rv;

rv = GetPartHandlerListForFileExt(fileExtension);

```

---

## GetPartHandlerListForFileExt (OS/2) Parameter - fileExtension

**fileExtension** ([string](#)) - input

The file extension that this part handler can edit. This is an optional parameter and can be set to KODNULL.

---

## GetPartHandlerListForFileExt (OS/2) Return Value - rv

**rv** ([\\_IDL\\_SEQUENCE\\_string](#)) - returns

A list of part handlers for the file extension. If the file extension is not specified, a list of all part handlers is returned.

When it is finished using this value, the client code is responsible for freeing each object in the sequence (by using SOMFree) and the sequence buffer.

---

## GetPartHandlerListForFileExt (OS/2) - Parameters

**fileExtension** ([string](#)) - input

The file extension that this part handler can edit. This is an optional parameter and can be set to KODNULL.

**rv** ([\\_IDL\\_SEQUENCE\\_string](#)) - returns

A list of part handlers for the file extension. If the file extension is not specified, a list of all part handlers is returned.

When it is finished using this value, the client code is responsible for freeing each object in the sequence (by using SOMFree) and the sequence buffer.

---

## GetPartHandlerListForFileExt (OS/2) - Topics

**Class:**

ODPartHandlerRegistry

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

---

## GetPartHandlerListForFileTypes (OS/2)

---

## GetPartHandlerListForFileTypes (OS/2) - Syntax

This method returns a list of part-handler names for the specified file type.

```
#define INCL_ODPARTHANDLERREGISTRY
#define INCL_ODAPI
#include <os2.h>

string          fileType;
_IDL_SEQUENCE_string  rv;

rv = GetPartHandlerListForFileTypes(fileType);
```

---

## GetPartHandlerListForFileTypes (OS/2) Parameter - fileType

**fileType** ([string](#)) - input

The file type that can be edited by the part handlers. This is an optional parameter and can be set to KODNULL.

---

## GetPartHandlerListForFileTypes (OS/2) Return Value - rv

**rv** ([\\_IDL\\_SEQUENCE\\_string](#)) - returns

A list of part handlers for the specified file type. If the file type is not specified, a list of all part handlers is returned.

When it is finished using this parameter, the client code is responsible for freeing each object in the sequence (using SOMFree) and the sequence buffer.

---

## GetPartHandlerListForFileTypes (OS/2) - Parameters

**fileType** ([string](#)) - input

The file type that can be edited by the part handlers. This is an optional parameter and can be set to KODNULL.

**rv** ([\\_IDL\\_SEQUENCE\\_string](#)) - returns

A list of part handlers for the specified file type. If the file type is not specified, a list of all part handlers is returned.

When it is finished using this parameter, the client code is responsible for freeing each object in the sequence (using SOMFree) and the sequence buffer.

---

## GetPartHandlerListForFileTypes (OS/2) - Topics

**Class:**  
ODPartHandlerRegistry

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## GetPartKindList (OS/2)

---

### GetPartKindList (OS/2) - Syntax

This method returns a list of part kinds using part handler names and categories as search criteria.

```
#include <os2.h>

ISOSString      partHandlerName;
string          category;
_IDL_SEQUENCE_string rv;

rv = GetPartKindList(partHandlerName, category);
```

---

### GetPartKindList (OS/2) Parameter - partHandlerName

**partHandlerName** ([ISOSString](#)) - input

The name of the part handler that can edit the part kinds. This parameter is optional and can be set to KODNULL.

---

### GetPartKindList (OS/2) Parameter - category

**category** ([string](#)) - input

A category to which the part kinds belong. This parameter is optional and can be set to KODNULL.

---

### GetPartKindList (OS/2) Return Value - rv

**rv** ([\\_IDL\\_SEQUENCE\\_string](#)) - returns

A list of part kinds that meet the search criteria. If the part handler and category are not specified, a list of all registered part kinds is returned.

When it is finished using this value, the client code is responsible for freeing each object in the sequence (by calling SOMFree) and the sequence buffer.



---

## GetPartKindList (OS/2) - Parameters

**partHandlerName** ([ISOStrng](#)) - input

The name of the part handler that can edit the part kinds. This parameter is optional and can be set to KODNULL.

**category** ([string](#)) - input

A category to which the part kinds belong. This parameter is optional and can be set to KODNULL.

**rv** ([\\_IDL\\_SEQUENCE\\_string](#)) - returns

A list of part kinds that meet the search criteria. If the part handler and category are not specified, a list of all registered part kinds is returned.

When it is finished using this value, the client code is responsible for freeing each object in the sequence (by calling SOMFree) and the sequence buffer.

---

## GetPartKindList (OS/2) - Topics

### Class:

ODPartHandlerRegistry

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

---

## GetPreferredPartHandler (OS/2)

---

## GetPreferredPartHandler (OS/2) - Syntax

This method returns the preferred part-handler name for the specified part kind.

```
#define INCL_ODPARTHANDLERREGISTRY
#define INCL_ODAPI
#include <os2.h>
```

```
ISOStrng    partKindName;
ISOStrng    rv;
```

```
rv = GetPreferredPartHandler(partKindName);
```

---

## GetPreferredPartHandler (OS/2) Parameter - partKindName

**partKindName** ([ISOString](#)) - input  
The part kind name.

---

## GetPreferredPartHandler (OS/2) Return Value - rv

**rv** ([ISOString](#)) - returns  
The preferred name of the part handler. When it is finished using this value, the client code is responsible for freeing each object in the sequence (by calling SOMFree) and the sequence buffer.

---

## GetPreferredPartHandler (OS/2) - Parameters

**partKindName** ([ISOString](#)) - input  
The part kind name.

**rv** ([ISOString](#)) - returns  
The preferred name of the part handler. When it is finished using this value, the client code is responsible for freeing each object in the sequence (by calling SOMFree) and the sequence buffer.

---

## GetPreferredPartHandler (OS/2) - Topics

**Class:**  
ODPartHandlerRegistry

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## GetPreferredPartHandlerForCategory (OS/2)

---

## GetPreferredPartHandlerForCategory (OS/2) - Syntax

This method returns the preferred part-handler name for the category.

```
#define INCL_ODPARTHANDLERREGISTRY
#define INCL_ODAPI
#include <os2.h>
```

```
ISOString    category;
ISOString    rv;
```

```
rv = GetPreferredPartHandlerForCategory(category);
```

---

## GetPreferredPartHandlerForCategory (OS/2) Parameter - category

**category** ([ISOString](#)) - input  
The category of the part.

---

## GetPreferredPartHandlerForCategory (OS/2) Return Value - rv

**rv** ([ISOString](#)) - returns  
The preferred part handler name.

---

## GetPreferredPartHandlerForCategory (OS/2) - Parameters

**category** ([ISOString](#)) - input  
The category of the part.

**rv** ([ISOString](#)) - returns  
The preferred part handler name.

---

## GetPreferredPartHandlerForCategory (OS/2) - Topics

**Class:**  
ODPartHandlerRegistry

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## GetPreferredPartHandlerForFileExt (OS/2)

---

## GetPreferredPartHandlerForFileExt (OS/2) - Syntax

This method returns the preferred part-handler name for the specified file extension.

```
#define INCL_ODPARTHANDLERREGISTRY
#define INCL_ODAPI
#include <os2.h>

ISOSString    partFileExtensionName;
ISOSString    rv;

rv = GetPreferredPartHandlerForFileExt (partFileExtensionName);
```

---

## GetPreferredPartHandlerForFileExt (OS/2) Parameter - partFileExtensionName

**partFileExtensionName** ([ISOSString](#)) - input  
The file extension of the part.

---

## GetPreferredPartHandlerForFileExt (OS/2) Return Value - rv

**rv** ([ISOSString](#)) - returns  
The preferred name of the part handler. When it is finished using this value, the client code is responsible for freeing it by calling SOMFree.

---

## GetPreferredPartHandlerForFileExt (OS/2) - Parameters

**partFileExtensionName** ([ISOSString](#)) - input  
The file extension of the part.

**rv** ([ISOSString](#)) - returns  
The preferred name of the part handler. When it is finished using this value, the client code is responsible for freeing it by calling SOMFree.

---

## GetPreferredPartHandlerForFileExt (OS/2) - Topics

**Class:**  
ODPartHandlerRegistry

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## GetPreferredPartHandlerForFileType (OS/2)

---

## GetPreferredPartHandlerForFileType (OS/2) - Syntax

This method returns the preferred part-handler name for the specified file type.

```
#define INCL_ODPARTHANDLERREGISTRY
#define INCL_ODAPI
#include <os2.h>

ISOString    partFileName;
ISOString    rv;

rv = GetPreferredPartHandlerForFileType(partFileName);
```

---

## GetPreferredPartHandlerForFileType (OS/2) Parameter - partFileTy

**partFileName** ([ISOString](#)) - input  
The file type of the part.

---

## GetPreferredPartHandlerForFileType (OS/2) Return Value - rv

**rv** ([ISOString](#)) - returns  
The preferred name of the part handler. When it is finished using this value, the client code is responsible for freeing each object in the sequence (by calling SOMFree) and the sequence buffer.

---

## GetPreferredPartHandlerForFileType (OS/2) - Parameters

**partFileName** ([ISOString](#)) - input  
The file type of the part.

**rv** ([ISOString](#)) - returns  
The preferred name of the part handler. When it is finished using this value, the client code is responsible for freeing each object in the sequence (by calling SOMFree) and the sequence buffer.

---

## GetPreferredPartHandlerForFileType (OS/2) - Topics

**Class:**  
ODPartHandlerRegistry

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

---

## RegisterPartHandlerClass (OS/2)

---

### RegisterPartHandlerClass (OS/2) - Syntax

This method registers a part handler library with the OpenDoc registry.

```
#define INCL_ODPARTHANDLERREGISTRY
#define INCL_ODAPI
#include <os2.h>

string  className;
string  DLLName;
long    cTemplate;
long    reserved;
long    rv;          /* Return codes. */

rv = RegisterPartHandlerClass(className, DLLName,
                             cTemplate, reserved);
```

---

### RegisterPartHandlerClass (OS/2) Parameter - className

**className** ([string](#)) - input  
The name of the SOM class to be registered.

---

### RegisterPartHandlerClass (OS/2) Parameter - DLLName

**DLLName** ([string](#)) - input  
A qualified or unqualified name of the DLL for the class.

---

### RegisterPartHandlerClass (OS/2) Parameter - cTemplate

**cTemplate** (long) - input  
A flag indicating whether an OpenDoc template should be created for each part kind that this handler supports

---

## RegisterPartHandlerClass (OS/2) Parameter - reserved

**reserved** (long) - input  
Reserved value.

---

## RegisterPartHandlerClass (OS/2) Return Value - rv

**rv** (long) - returns  
Return codes.

NO\_ERROR

ERROR\_COULD\_NOT\_LOAD\_DLL

ERROR\_COUNDT\_NOT\_FIND\_CLASS\_IN\_DLL

ERROR\_INVALID\_PART\_KIND

---

## RegisterPartHandlerClass (OS/2) - Parameters

**className** (string) - input  
The name of the SOM class to be registered.

**DLLName** (string) - input  
A qualified or unqualified name of the DLL for the class.

**cTemplate** (long) - input  
A flag indicating whether an OpenDoc template should be created for each part kind that this handler supports

**reserved** (long) - input  
Reserved value.

**rv** (long) - returns  
Return codes.

NO\_ERROR

ERROR\_COULD\_NOT\_LOAD\_DLL

ERROR\_COUNDT\_NOT\_FIND\_CLASS\_IN\_DLL

ERROR\_INVALID\_PART\_KIND

---

## RegisterPartHandlerClass (OS/2) - Remarks

This method creates an instance of the specified class using the specified DLL name. The class is then queried for its registration information. The successful completion of this method depends on the information that is provided by the part-handler class. If the handler has already

been registered, this method replaces the existing handler information.

---

## RegisterPartHandlerClass (OS/2) - Topics

### Class:

ODPartHandlerRegistry

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

---

## SetPreferredPartHandler (OS/2)

---

## SetPreferredPartHandler (OS/2) - Syntax

This method sets the preferred part-handler name.

```
#define INCL_ODPARTHANDLERREGISTRY
#define INCL_ODAPI
#include <os2.h>

ISOString    partKindName;
ISOString    partHandlerName;
long         rv;          /* Return codes. */

rv = SetPreferredPartHandler(partKindName,
                             partHandlerName);
```

---

## SetPreferredPartHandler (OS/2) Parameter - partKindName

**partKindName** ([ISOString](#)) - input  
The part kind name.

---

## SetPreferredPartHandler (OS/2) Parameter - partHandlerName

**partHandlerName** ([ISOString](#)) - input  
The preferred part handler name.

---



# SetPreferredPartHandler (OS/2) Return Value - rv

**rv** (long) - returns  
Return codes.

NO\_ERROR

ERROR\_COULD\_NOT\_FIND\_PART\_KIND

---

## SetPreferredPartHandler (OS/2) - Parameters

**partKindName** ([ISOString](#)) - input  
The part kind name.

**partHandlerName** ([ISOString](#)) - input  
The preferred part handler name.

**rv** (long) - returns  
Return codes.

NO\_ERROR

ERROR\_COULD\_NOT\_FIND\_PART\_KIND

---

## SetPreferredPartHandler (OS/2) - Topics

**Class:**  
ODPartHandlerRegistry

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## SetPreferredPartHandlerForCategory (OS/2)

---

## SetPreferredPartHandlerForCategory (OS/2) - Syntax

This method sets the preferred part-handler name for the category.

```
#define INCL_ODPARTHANDLERREGISTRY
```

```

#define INCL_ODAPI
#include <os2.h>

ISOString    category;
ISOString    partHandlerName;
long         rv;          /* Return codes. */

rv = SetPreferredPartHandlerForCategory(category,
    partHandlerName);

```

-----

## SetPreferredPartHandlerForCategory (OS/2) Parameter - category

**category** ([ISOString](#)) - input  
The category of the part.

-----

## SetPreferredPartHandlerForCategory (OS/2) Parameter - partHandl

**partHandlerName** ([ISOString](#)) - input  
The preferred part handler name.

-----

## SetPreferredPartHandlerForCategory (OS/2) Return Value - rv

**rv** (long) - returns  
Return codes.  
  
NO\_ERROR

-----

## SetPreferredPartHandlerForCategory (OS/2) - Parameters

**category** ([ISOString](#)) - input  
The category of the part.

**partHandlerName** ([ISOString](#)) - input  
The preferred part handler name.

**rv** (long) - returns  
Return codes.  
  
NO\_ERROR

-----

## SetPreferredPartHandlerForCategory (OS/2) - Topics

**Class:**

ODPartHandlerRegistry

Select an item:

[Syntax](#)[Parameters](#)[Returns](#)

---

## SetPreferredPartHandlerForFileExt (OS/2)

---

### SetPreferredPartHandlerForFileExt (OS/2) - Syntax

This method sets the preferred part-handler name for the specified file extension.

```
#define INCL_ODPARTHANDLERREGISTRY
#define INCL_ODAPI
#include <os2.h>

ISOSString    partFileExtensionName;
ISOSString    partHandlerName;
long          rv;                                /* Return code. */

rv = SetPreferredPartHandlerForFileExt(partFileExtensionName,
                                       partHandlerName);
```

---

### SetPreferredPartHandlerForFileExt (OS/2) Parameter - partFileExt

**partFileExtensionName** ([ISOSString](#)) - input  
The file extension of the part.

---

### SetPreferredPartHandlerForFileExt (OS/2) Parameter - partHandler

**partHandlerName** ([ISOSString](#)) - input  
The preferred name of the part handler.

---

### SetPreferredPartHandlerForFileExt (OS/2) Return Value - rv

**rv** (long) - returns

Return code.

NO\_ERROR

ERROR\_COULD\_NOT\_FIND\_PART\_KIND

-----

## SetPreferredPartHandlerForFileExt (OS/2) - Parameters

**partFileExtensionName** ([ISOString](#)) - input  
The file extension of the part.

**partHandlerName** ([ISOString](#)) - input  
The preferred name of the part handler.

**rv** (long) - returns  
Return code.

NO\_ERROR

ERROR\_COULD\_NOT\_FIND\_PART\_KIND

-----

## SetPreferredPartHandlerForFileExt (OS/2) - Topics

**Class:**  
ODPartHandlerRegistry

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)

-----

## SetPreferredPartHandlerForFileType (OS/2)

-----

## SetPreferredPartHandlerForFileType (OS/2) - Syntax

This method sets the preferred part-handler name for the specified file type.

```
#define INCL_ODPARTHANDLERREGISTRY
#define INCL_ODAPI
#include <os2.h>

ISOString    partFileName;
ISOString    partHandlerName;
long         rv;          /* Return code. */

rv = SetPreferredPartHandlerForFileType(partFileName,
    partHandlerName);
```

---

## SetPreferredPartHandlerForFileType (OS/2) Parameter - partFileType

**partFileName** ([ISOString](#)) - input  
The file type of the part.

---

## SetPreferredPartHandlerForFileType (OS/2) Parameter - partHandlerName

**partHandlerName** ([ISOString](#)) - input  
The preferred name of the part handler.

---

## SetPreferredPartHandlerForFileType (OS/2) Return Value - rv

**rv** (long) - returns  
Return code.

NO\_ERROR

ERROR\_COULD\_NOT\_FIND\_PART\_KIND

---

## SetPreferredPartHandlerForFileType (OS/2) - Parameters

**partFileName** ([ISOString](#)) - input  
The file type of the part.

**partHandlerName** ([ISOString](#)) - input  
The preferred name of the part handler.

**rv** (long) - returns  
Return code.

NO\_ERROR

ERROR\_COULD\_NOT\_FIND\_PART\_KIND

---

## SetPreferredPartHandlerForFileType (OS/2) - Topics

**Class:**

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

# ODPersistentObject

**Class Definition File:** PSTOBJ.IDL

## Class Hierarchy

```

SOMObject
├── ODOObject
│   ├── ODRefCntObject
│   └── ODPersistentObject

```

## Description

An object of the ODPersistentObject class implements the protocol for saving and restoring objects to persistent storage.

Certain objects created in one OpenDoc session can be saved to persistent storage; in a later session, the same objects can be recreated from their stored data. Similarly, an object that is not currently being used can be saved to persistent storage to free space in memory; if the object is needed later in the same session, it can be recreated from its stored data. An object whose state can be saved to persistent storage is called a *persistent object*.

The ODPersistentObject class is the abstract superclass for all OpenDoc classes whose objects need to be saved persistently. You should not create immediate subclasses or instances of ODPersistentObject itself. Three of its subclasses are concrete classes: [ODFrame](#), [ODLink](#), and [ODLinkSource](#). You can create instances of those classes, but you should not create subclasses of them. The ODPersistentObject class has a fourth subclass, [ODPart](#), which is an abstract class. To develop a part editor, you need to create a subclass of [ODPart](#). The persistent objects that can be saved and recreated from storage are frames, links, link sources, and parts.

Each persistent object has an associated storage unit in which it can store its data persistently. Within each draft in a particular session, the storage unit is given a unique identifier (ID) that designates both the storage unit itself and the persistent object whose data it contains. This ID is not part of the object's persistent data but is valid only for the duration of the session.

The process of storing a persistent object's data is called *externalizing* the object. When a persistent object writes itself to storage, it writes into its storage unit whatever information is necessary to restore it to its current state. When a stored persistent object is recreated, it initializes itself to its previous state by reading the data it stored when it was last written.

A persistent object can also be *cloned*, that is, the object and all additional objects that it references can be copied. Typically, an object is cloned whenever its data must be transferred, for example, when it is copied to the clipboard.

For more information about storage units, see the class description of [ODStorageUnit](#). For more information about cloning, see the chapter on data transfer in the *OpenDoc Programming Guide*. For more information about the different kinds of persistent objects, see the class descriptions for [ODFrame](#), [ODLink](#), [ODLinkSource](#), and [ODPart](#).

## Methods

The methods defined by the ODPersistentObject class include:

- [CloneInto](#)
- [Externalize](#)
- [GetID](#)
- [GetStorageUnit](#)
- [InitPersistentObject](#)
- [InitPersistentObjectFromStorage](#)
- [ReleaseAll](#)

## Overridden Methods

There are currently no methods overridden by the ODPersistentObject class.

---

# CloneInto

---

## CloneInto - Syntax

This method should clone this persistent object by copying its data into a specified storage unit.

```
#define INCL_ODPERSISTENTOBJECT
#define INCL_ODAPI
#include <os2.h>

ODDraftKey      key;
ODStorageUnit   *toSU;
ODFrame         *scope;

CloneInto(key, toSU, scope);
```

---

## CloneInto Parameter - key

**key** (ODDraftKey) - input

The draft key identifying the current cloning operation. The key provides thread-safe access to cloning.

---

## CloneInto Parameter - toSU

**toSU** (ODStorageUnit \*) - input

A reference to the destination storage unit to which the data is to be copied.

---

## CloneInto Parameter - scope

**scope** (ODFrame \*) - input

A reference to a frame object that defines the scope of the cloning operation or KODNULL if all referenced objects are within the scope.

---

## CloneInto - Return Value

None.

---

## CloneInto - Parameters

**key** ([ODDraftKey](#)) - input

The draft key identifying the current cloning operation. The key provides thread-safe access to cloning.

**toSU** ([ODStorageUnit](#) \*) - input

A reference to the destination storage unit to which the data is to be copied.

**scope** ([ODFrame](#) \*) - input

A reference to a frame object that defines the scope of the cloning operation or [kODNULL](#) if all referenced objects are within the scope.

None.

---

## CloneInto - Remarks

Your part should never call this method directly; it is called by the draft's [Clone](#) and [WeakClone](#) methods.

The *scope* parameter determines which of the referenced objects are within the scope of this cloning operation. Typically, the *scope* parameter is a reference to a frame and only those objects embedded in that frame are within the scope.

This method copies this object's data into the specified destination storage unit and clones any additional objects to which this object has strong and weak persistent references and that are within the scope of this cloning operation. Objects referenced by strong persistent references are strongly cloned by recursive calls to the [Clone formt=extonly](#) method; objects referenced by weak persistent references are weakly cloned by calls to the [WeakClone formt=extonly](#) method. Otherwise, only those objects logically enclosed in the specified frame are within the scope and should be cloned.

---

## CloneInto - Override Policy

If you subclass [ODPersistentObject](#), you must override this method to support data interchange of internal data. Your override method must call its inherited method at the beginning of its implementation.

---

## CloneInto - Related Methods

### Related Methods

- [ODDraft::Clone](#)
  - [ODDraft::WeakClone](#)
- 

## CloneInto - Topics

### Class:

[ODPersistentObject](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Related Methods](#)



---

# Externalize

---

## Externalize - Syntax

This method should store the data needed to restore this persistent object to its current state.

```
#define INCL_ODPERSISTENTOBJECT
#define INCL_ODAPI
#include <os2.h>
```

```
Externalize();
```

---

## Externalize - Return Value

None.

---

## Externalize - Parameters

None.

---

## Externalize - Remarks

Your part calls this method of a persistent object (for example, a part or frame) to write that object's data to persistent storage. This method writes the object's persistent state to the properties and values in the object's storage unit. It must write sufficient information to allow the object's *InitClass FromStorage* method to restore the object to its current state; for example, a part's *Externalize* method must write the information needed by its [InitPartFromStorage](#) method.

---

## Externalize - Override Policy

If you subclass of [ODPersistentObject](#), you must override the method if your part maintains persistent content. Your override method must call its inherited method at the beginning of its implementation.

---

# Externalize - Related Methods

## Related Methods

- [ODPersistentObject::InitPersistentObject](#)
  - [ODPersistentObject::InitPersistentObjectFromStorage](#)
- 

# Externalize - Topics

## Class:

[ODPersistentObject](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Override Policy](#)  
[Related Methods](#)

---

# GetID

---

## GetID - Syntax

This method returns the unique ID of this persistent object for the current draft in the current session.

```
#define INCL_ODPERSISTENTOBJECT
#define INCL_ODAPI
#include <os2.h>

ODID    rv;

rv = GetID();
```

---

## GetID Return Value - rv

**rv** ([ODID](#)) - returns  
The unique ID of this persistent object.

---

# GetID - Parameters

**rv** ([ODID](#)) - returns  
The unique ID of this persistent object.

-----

# GetID - Remarks

The ID of the persistent object is also the ID of its storage unit. The ID is not persistent data. This persistent object can have a different ID in each different session and in each different draft within the same session.

-----

# GetID - Topics

**Class:**  
[ODPersistentObject](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

-----

# GetStorageUnit

-----

# GetStorageUnit - Syntax

This method returns a reference to the storage unit in which this persistent object stores its data.

```
#define INCL_ODPERSISTENTOBJECT
#define INCL_ODAPI
#include <os2.h>
```

```
ODStorageUnit      *rv;
```

```
rv = GetStorageUnit();
```

-----

# GetStorageUnit Return Value - rv

**rv** (ODStorageUnit \*) - returns

A reference to the storage unit of this persistent object or KODNULL if the object does not have a storage unit.

---

## GetStorageUnit - Parameters

**rv** (ODStorageUnit \*) - returns

A reference to the storage unit of this persistent object or KODNULL if the object does not have a storage unit.

---

## GetStorageUnit - Remarks

Whenever your part calls this method, it should check that the return value is not null before attempting to use the storage unit because nonpersistent frames, which do not have storage units, may be added to your part.

You should never cache a reference to the returned storage unit; instead, you must call this method whenever you access the storage unit. This method does not increment the reference count of the returned storage unit.

---

## GetStorageUnit - Topics

### Class:

ODPersistentObject

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

---

## InitPersistentObject

---

## InitPersistentObject - Syntax

This method initializes this newly created persistent object.

```
#define INCL_ODPERSISTENTOBJECT
#define INCL_ODAPI
#include <os2.h>

ODStorageUnit      *storageUnit;

InitPersistentObject(storageUnit);
```

---

# InitPersistentObject Parameter - storageUnit

**storageUnit** (ODStorageUnit \*) - input  
A reference to a storage unit of this persistent object.

---

## InitPersistentObject - Return Value

None.

---

## InitPersistentObject - Parameters

**storageUnit** (ODStorageUnit \*) - input  
A reference to a storage unit of this persistent object.

---

None.

---

## InitPersistentObject - Remarks

Your part editor should never call this method directly; it is called automatically whenever a persistent object is created for the first time. Every existing OpenDoc subclass of [ODPersistentObject](#) (including [ODPart](#)) has an initialization method that calls the inherited `InitPersistentObject` method at the beginning of its implementation.

For example, the initialization method of [ODPart](#) is the [InitPart](#) method. When you call the draft's [CreatePart](#) method to create a part of your class, that factory method calls your part's [InitPart](#) method. Your part's [InitPart](#) method should call the inherited [InitPart](#) method, which calls the `InitPersistentObject` method.

The `InitPersistentObject` method is not called when a stored object is recreated; instead, the [InitPersistentObjectFromStorage](#) method is called to restore the object to its state at the time it was last saved.

---

## InitPersistentObject - Related Methods

### Related Methods

- [ODPart::InitPart](#)
  - [ODPersistentObject::Externalize](#)
  - [ODPersistentObject::InitPersistentObjectFromStorage](#)
- 

## InitPersistentObject - Topics

**Class:**

ODPersistentObject

Select an item:

[Syntax](#)[Parameters](#)[Returns](#)[Remarks](#)[Related Methods](#)

---

## InitPersistentObjectFromStorage

---

### InitPersistentObjectFromStorage - Syntax

This method initializes this recreated persistent object from its stored data.

```
#define INCL_ODPERSISTENTOBJECT
#define INCL_ODAPI
#include <os2.h>

ODStorageUnit      *storageUnit;

InitPersistentObjectFromStorage(storageUnit);
```

---

### InitPersistentObjectFromStorage Parameter - storageUnit

**storageUnit** (ODStorageUnit \*) - input

A reference to a storage unit of this persistent object.

---

### InitPersistentObjectFromStorage - Return Value

None.

---

### InitPersistentObjectFromStorage - Parameters

**storageUnit** (ODStorageUnit \*) - input

A reference to a storage unit of this persistent object.

None.

---

## InitPersistentObjectFromStorage - Remarks

Your part editor should never call this method directly; it is called automatically whenever a persistent object is recreated from stored data. Every existing OpenDoc subclass of [ODPersistentObject](#) (including [ODPart](#)) has an *InitClass FromStorage* method to initialize an object of that class from its stored data; that method calls the inherited *InitPersistentObjectFromStorage* method and then reads any data that was previously written by the [Externalize](#) method when the object was stored. The object's factory method calls the object's *InitClass FromStorage* method when recreating the object from its stored data.

For example, the [ODFrame](#) class has a private *InitClass FromStorage* method. When you call the draft's [AcquireFrame](#) method to recreate a frame, that factory method calls the recreated frame's *InitFrameFromStorage* method.

---

## InitPersistentObjectFromStorage - Related Methods

### Related Methods

- [ODPart::InitPartFromStorage](#)
- [ODPersistentObject::Externalize](#)
- [ODPersistentObject::InitPersistentObject](#)

---

## InitPersistentObjectFromStorage - Topics

### Class:

[ODPersistentObject](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

## ReleaseAll

---

## ReleaseAll - Syntax

This method should release all transitory references from this persistent object to other reference-counted objects.

```
#define INCL_ODPERSISTENTOBJECT
#define INCL_ODAPI
```

```
#include <os2.h>
```

```
ReleaseAll();
```

---

## ReleaseAll - Return Value

None.

---

## ReleaseAll - Parameters

None.

---

## ReleaseAll - Remarks

The factory object that creates each persistent object calls that object's `ReleaseAll` method before deleting it.

---

## ReleaseAll - Override Policy

If you subclass [ODPersistentObject](#), you must override this method if its objects maintains references to other persistent objects. Your override method must call its inherited method at the end of its implementation.

---

## ReleaseAll - Topics

### Class:

[ODPersistentObject](#)

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Override Policy](#)

---

## ODPlatformCanvas

**Class Definition File:** PLATCANV.IDL



## Class Hierarchy

SOMObject  
ODObject  
    **ODPlatformCanvas**

## Description

Instances of the ODPlatformCanvas are used to initialize offscreen static, offscreen, dynamic, and onscreen static canvases. Instances of the derived class, ODPlatformWindowCanvas, are used to initialize onscreen dynamic canvases; however, because ODPlatformWindowCanvas introduces no new methods, objects of both classes are referred to using the base class pointer.

## Methods

The methods defined by the ODPlatformCanvas class include:

- [GetPS](#)
- [GetWindow](#)
- [HasWindow](#)
- [ReleasePS](#)
- [SetPS](#)

## Overridden Methods

There are currently no methods overridden by the ODPlatformCanvas class.

---

# GetPS (OS/2)

---

## GetPS (OS/2) - Syntax

This method returns the handle to the presentation space (PS) of the specified facet.

```
#define INCL_ODPLATFORMCANVAS
#define INCL_ODAPI
#include <os2.h>

ODFacet      *facet;
HPS          rv;

rv = GetPS(facet);
```

---

## GetPS (OS/2) Parameter - facet

**facet** (ODFacet \*) - input

The facet for which the PS is being obtained. This parameter is ignored by the [ODPlatformCanvas](#) class. For the derived class [ODPlatformWindowCanvas](#), it is used to obtain a PS for the specified facet's facet window.

---

## GetPS (OS/2) Return Value - rv

**rv** ([HPS](#)) - returns  
A handle to a GPI PS specified in the call to the facet's [CreatePlatformCanvas](#) method.

---

## GetPS (OS/2) - Parameters

**facet** ([ODFacet \\*](#)) - input  
The facet for which the PS is being obtained. This parameter is ignored by the [ODPlatformCanvas](#) class. For the derived class [ODPlatformWindowCanvas](#), it is used to obtain a PS for the specified facet's facet window.

**rv** ([HPS](#)) - returns  
A handle to a GPI PS specified in the call to the facet's [CreatePlatformCanvas](#) method.

---

## GetPS (OS/2) - Topics

**Class:**  
[ODPlatformCanvas](#)

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## GetWindow (OS/2)

---

## GetWindow (OS/2) - Syntax

This method is overridden and implemented by the derived [ODPlatformWindowCanvas](#) class to return the window object specified when this object was created. If this method is called on an instance of this object type, a [KODerrInvalidObjectType](#) exception is set.

```
#define INCL_ODPLATFORMCANVAS
#define INCL_ODAPI
#include <os2.h>

ODWindow      *rv;

rv = GetWindow();
```

---

## GetWindow (OS/2) Return Value - rv

**rv** ([ODWindow \\*](#)) - returns  
This method returns [KODNULL](#) and sets the [KODerrInvalidObjectType](#) exception.

---

## GetWindow (OS/2) - Parameters

**rv** (ODWindow \*) - returns  
This method returns KODNULL and sets the KODerrInvalidObjectType exception.

---

## GetWindow (OS/2) - Exception Handling

KODerrInvalidObjectType This method should not be called on an instance of this object.

---

## GetWindow (OS/2) - Topics

**Class:**  
ODPlatformCanvas

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Exception Handling](#)

---

## HasWindow (OS/2)

---

## HasWindow (OS/2) - Syntax

This method indicates whether a given [ODPlatformCanvas](#) reference refers to an instance of the [ODPlatformCanvas](#) or [ODPlatformWindowCanvas](#) class.

```
#define INCL_ODPLATFORMCANVAS
#define INCL_ODAPI
#include <os2.h>

ODBoolean    rv;

rv = HasWindow();
```

---

## HasWindow (OS/2) Return Value - rv

**rv** ([ODBoolean](#)) - returns  
This method returns the following value:

kODFalse      The object is an instance of the [ODPlatformCanvas](#) class.

---

## HasWindow (OS/2) - Parameters

**rv** ([ODBoolean](#)) - returns  
This method returns the following value:

kODFalse      The object is an instance of the [ODPlatformCanvas](#) class.

---

## HasWindow (OS/2) - Topics

**Class:**  
[ODPlatformCanvas](#)

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## ReleasePS (OS/2)

---

## ReleasePS (OS/2) - Syntax

This method performs no action. It is overridden and implemented by the derived class [ODPlatformWindowCanvas](#).

```
#define INCL_ODPLATFORMCANVAS
#define INCL_ODAPI
#include <os2.h>
```

```
ODFacet      *facet;
```

```
ReleasePS(facet);
```

---

## ReleasePS (OS/2) Parameter - facet

**facet** (ODFacet \*) - input  
The facet passed to the corresponding [GetPS](#) method.

---

## ReleasePS (OS/2) - Return Value

None.

---

## ReleasePS (OS/2) - Parameters

**facet** (ODFacet \*) - input  
The facet passed to the corresponding [GetPS](#) method.

None.

---

## ReleasePS (OS/2) - Topics

**Class:**  
ODPlatformCanvas

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## SetPS (OS/2)

---

## SetPS (OS/2) - Syntax

This method is overridden and implemented by the derived [ODPlatformWindowCanvas](#) class to allow a noncached presentation space to be associated with a facet window. If this method is called on an instance of this object type, a kODerrInvalidObjectType exception is set.

```
#define INCL_ODPLATFORMCANVAS
#define INCL_ODAPI
#include <os2.h>
```

```
HPS          hps;
ODFacet      *facet;
```

```
SetPS(hps, facet);
```

---

## SetPS (OS/2) Parameter - hps

**hps** ([HPS](#)) - input  
A handle to the presentation space.

---

## SetPS (OS/2) Parameter - facet

**facet** ([ODFacet \\*](#)) - input  
A pointer to an [ODFacet](#) object.

---

## SetPS (OS/2) - Return Value

None.

---

## SetPS (OS/2) - Parameters

**hps** ([HPS](#)) - input  
A handle to the presentation space.

**facet** ([ODFacet \\*](#)) - input  
A pointer to an [ODFacet](#) object.

None.

---

## SetPS (OS/2) - Exception Handling

[KODerrInvalidObjectType](#)

This method should not be called on an instance of this object.

---

## SetPS (OS/2) - Topics

**Class:**  
ODPlatformCanvas

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Exception Handling](#)

---

# ODPlatformTypeList

**Class Definition File:** PFTYPLS.IDL

## Class Hierarchy

SOMObject  
ODObject  
    **ODPlatformTypeList**

## Description

An object of the ODPlatformTypeList class is an ordered set of [ODPlatformType](#) elements.

A platform type list is an ordered set of elements, each specifying a different value of some platform-specific type. Because these elements are of the [ODPlatformType](#) type, they can represent any platform-specific value used to identify data formats for data interchange.

To create a platform type list, call the [CreatePlatformTypeList](#) method of the storage-system object. If the call to that method specifies an existing platform type list, the new platform type list is initialized to a copy of that list; otherwise, the new list is initialized to an empty list (a list with no elements).

You can add elements one at a time to the end of the platform type list. OpenDoc ensures that each element of a platform type list is unique; if you attempt to add a value that is already in the list, the list remains unchanged. You can remove elements from the list, test whether the list contains a particular element, and get the number of elements in the list. If you need to perform an operation for each element of the list, you can create an object of the [ODPlatformTypeListIterator](#) class and use it to iterate through the list.

## Methods

The methods defined by the ODPlatformTypeList class include:

- [AddLast](#)
- [Contains](#)
- [Count](#)
- [CreatePlatformTypeListIterator](#)
- [Remove](#)

## Overridden Methods

There are currently no methods overridden by the ODPlatformTypeList class.

---

# AddLast

---

## AddLast - Syntax

This method adds an element to the end of this platform type list.

```
#define INCL_ODPLATFORMTYPELIST
#define INCL_ODAPI
```

```
#include <os2.h>

ODPlatformType    type;

AddLast (type);
```

---

## AddLast Parameter - type

**type** (ODPlatformType) - input  
The element to be added to the list.

---

## AddLast - Return Value

None.

---

## AddLast - Parameters

**type** (ODPlatformType) - input  
The element to be added to the list.

None.

---

## AddLast - Remarks

If this platform type list already contains the specified element, no action is taken; otherwise, the specified element is added to the end of the list.

---

## AddLast - Exception Handling

KODerOutOfMemory

There is not enough memory to add the specified element to this platform type list.

---

## AddLast - Related Methods



## Related Methods

- [ODPlatformTypeList::Contains](#)
- [ODPlatformTypeList::Remove](#)

---

# AddLast - Topics

## Class:

ODPlatformTypeList

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

[Related Methods](#)

---

# Contains

---

## Contains - Syntax

This method indicates whether this platform type list contains the specified element.

```
#define INCL_ODPLATFORMTYPELIST
#define INCL_ODAPI
#include <os2.h>

ODPlatformType    type;
ODBoolean         rv;

rv = Contains(type);
```

---

## Contains Parameter - type

**type** ([ODPlatformType](#)) - input

The element to be tested for inclusion in this list.

---

## Contains Return Value - rv

**rv** ([ODBoolean](#)) - returns  
A flag indicating whether this platform type list contains the specified element.

kODTrue	This platform type list contains the specified element.
kODFalse	This platform type list does not contain the specified element.

---

## Contains - Parameters

**type** ([ODPlatformType](#)) - input  
The element to be tested for inclusion in this list.

**rv** ([ODBoolean](#)) - returns  
A flag indicating whether this platform type list contains the specified element.

kODTrue	This platform type list contains the specified element.
kODFalse	This platform type list does not contain the specified element.

---

## Contains - Related Methods

### Related Methods

- [ODPlatformTypeList::AddLast](#)
  - [ODPlatformTypeList::Remove](#)
- 

## Contains - Topics

**Class:**  
[ODPlatformTypeList](#)

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Related Methods](#)

---

## Count

---

## Count - Syntax

This method returns the number of elements in this platform type list.

```
#define INCL_ODPLATFORMTYPELIST
#define INCL_ODAPI
#include <os2.h>

ODULong    rv;

rv = Count();
```

-----

## Count Return Value - rv

**rv** ([ODULong](#)) - returns  
The number of elements in this platform type list or 0 if the list list is empty.

-----

## Count - Parameters

**rv** ([ODULong](#)) - returns  
The number of elements in this platform type list or 0 if the list list is empty.

-----

## Count - Topics

**Class:**  
ODPlatformTypeList

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)

-----

## CreatePlatformTypeListIterator

-----

## CreatePlatformTypeListIterator - Syntax

This method creates a platform type-list iterator for this platform type list.

```
#define INCL_ODPLATFORMTYPELIST
#define INCL_ODAPI
#include <os2.h>
```

```
ODPlatformTypeListIterator    *rv;  
rv = CreatePlatformTypeListIterator();
```

---

## CreatePlatformTypeListIterator Return Value - rv

**rv** (ODPlatformTypeListIterator \*) - returns  
A reference to the newly created platform type-list iterator.

---

## CreatePlatformTypeListIterator - Parameters

**rv** (ODPlatformTypeListIterator \*) - returns  
A reference to the newly created platform type-list iterator.

---

## CreatePlatformTypeListIterator - Remarks

You call this method if you need to apply an operation to each element of this platform type list.

While you are using the returned platform type-list iterator, you must not modify this platform type list; in particular, you must not add or remove elements, and you must not delete this platform type list.

You must delete the returned platform type-list iterator when it is no longer needed.

---

## CreatePlatformTypeListIterator - Exception Handling

kODErrOutOfMemory

There is not enough memory to create the iterator.

---

## CreatePlatformTypeListIterator - Topics

### Class:

ODPlatformTypeList

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

---

# Remove

---

## Remove - Syntax

This method removes the specified element from this platform type list.

```
#define INCL_ODPLATFORMTYPELIST
#define INCL_ODAPI
#include <os2.h>

ODPlatformType    type;

Remove (type);
```

---

## Remove Parameter - type

**type** (ODPlatformType) - input  
The element to be removed.

---

## Remove - Return Value

None.

---

## Remove - Parameters

**type** (ODPlatformType) - input  
The element to be removed.

None.

---

## Remove - Remarks

If this platform type list does not contain the specified element, no action is taken.

---

# Remove - Related Methods

## Related Methods

- [ODPlatformTypeList::AddLast](#)
  - [ODPlatformTypeList::Contains](#)
- 

# Remove - Topics

## Class:

[ODPlatformTypeList](#)

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)  
[Related Methods](#)

---

# ODPlatformTypeListIterator

**Class Definition File:** PFTLITR.IDL

## Class Hierarchy

SOMObject  
  ODObject  
    **ODPlatformTypeListIterator**

## Description

An object of the ODPlatformTypeListIterator class provides access to each element of a platform type list.

You use a platform type-list iterator to apply an operation to each element of a platform type list.

Your part creates a platform type-list iterator object by calling the platform type list object's [CreatePlatformTypeListIterator](#) method, which returns a reference to a platform type-list iterator object. Platform type-list iterator objects are not currently required by any OpenDoc methods, but your part may wish to use these objects for internal use.

While you are using a platform type-list iterator, you should not modify or delete the platform type list that created it. You must postpone adding elements to or removing elements from the platform type list until after you have deleted the iterator.

For more information on accessing objects through iterators, see the chapter on OpenDoc run-time features in the *OpenDoc Programming Guide*.

## Methods

The methods defined by the ODPlatformTypeListIterator class include:

- [First](#)
- [IsNotComplete](#)
- [Next](#)

## Overridden Methods

There are currently no methods overridden by the ODPlatformTypeListIterator class.

---

# First

---

## First - Syntax

This method begins the iteration and returns the first element in the platform type list that created this platform type-list iterator.

```
#define INCL_ODPLATFORMTYPELIST
#define INCL_ODAPI
#include <os2.h>

ODPlatformType    rv;

rv = First();
```

---

## First Return Value - rv

**rv** ([ODPlatformType](#)) - returns  
The first element in the platform type list or KODNULL if the platform type list is empty.

---

## First - Parameters

**rv** ([ODPlatformType](#)) - returns  
The first element in the platform type list or KODNULL if the platform type list is empty.

---

## First - Remarks

Your part must call this method before calling this platform type-list iterator's [IsNotComplete](#) method for the first time. This method may be called multiple times; each time, it resets the iteration.

---

## First - Exception Handling

<b>KODErrIteratorOutOfSync</b>	The platform type list was modified while the iteration was in progress.
--------------------------------	--

---

# First - Topics

## Class:

ODPlatformTypeListIterator

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

---

## IsNotComplete

---

## IsNotComplete - Syntax

This method indicates whether the iteration is incomplete.

```
#define INCL_ODPLATFORMTYPELIST
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    rv;
```

```
rv = IsNotComplete();
```

---

## IsNotComplete Return Value - rv

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the iteration is incomplete.

kODTrue

The iteration is incomplete.

kODFalse

The iteration is complete.

---

## IsNotComplete - Parameters

**rv** ([ODBoolean](#)) - returns

A flag indicating whether the iteration is incomplete.

kODTrue

The iteration is incomplete.

kODFalse



The iteration is complete.

---

## IsNotComplete - Remarks

Your part calls this method to test whether more elements remain in the platform type list. This method returns `KODTrue` if the preceding call to the [First](#) or [Next](#) method found an element. This method returns `KODFalse` when you have examined all the elements (that is, when the previous call to [First](#) or [Next](#) returned `KODNULL`). If the platform type list that created this iterator is empty, this method always returns `KODFalse`.

---

## IsNotComplete - Exception Handling

`KODErrIteratorNotInitialized`

This method was called before calling either the [First](#) or [Next](#) method to begin the iteration.

`KODErrIteratorOutOfSync`

The platform type list was modified while the iteration was in progress.

---

## IsNotComplete - Topics

### Class:

`ODPlatformTypeListIterator`

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

---

## Next

---

## Next - Syntax

This method returns the next element in the platform type list that created this platform type-list iterator.

```
#define INCL_ODPLATFORMTYPELIST
#define INCL_ODAPI
#include <os2.h>
```

```
ODPlatformType    rv;
```

```
rv = Next();
```

---

## Next Return Value - rv

**rv** ([ODPlatformType](#)) - returns

The next element in the platform type list or KODNULL if you have reached the end of the platform type list.

---

## Next - Parameters

**rv** ([ODPlatformType](#)) - returns

The next element in the platform type list or KODNULL if you have reached the end of the platform type list.

---

## Next - Remarks

If your part calls this method before calling this platform type-list iterator's [First](#) method to begin the iteration, then this method works the same as calling the [First](#) method.

---

## Next - Exception Handling

KODErrIteratorOutOfSync

The platform type list was modified while the iteration was in progress.

---

## Next - Topics

### Class:

ODPlatformTypeListIterator

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

---

## ODPlatformWindowCanvas

**Class Definition File:** WNDPCANV.IDL

### Class Hierarchy

SOMObject

ODObject

## Description

An object of the ODPlatformWindowCanvas class provides methods for setting and obtaining presentation spaces for drawing your part's content onto a window canvas. This class introduces no new methods. It overrides and implements methods introduced by the [ODPlatformCanvas](#) class. Objects of the ODPlatformWindowCanvas class are generally referred to using a reference to a [ODPlatformCanvas](#) object.

Instances of this object are created by OpenDoc for the root facet of a document window using the facet's [CreatePlatformWindowCanvas](#) method. In general, parts should not create instances of this class.

## Methods

There are currently no methods defined by the ODPlatformWindowCanvas class.

## Overridden Methods

The methods overridden by the ODPlatformWindowCanvas class include:

- [ODPlatformCanvas](#)
  - [GetPS](#)
  - [GetWindow](#)
  - [HasWindow](#)
  - [ReleasePS](#)
  - [SetPS](#)

-----

## GetPS (OS/2)

-----

## GetPS (OS/2) - Syntax

This method returns the handle to the presentation space (PS) of the specified facet.

```
#define INCL_ODPLATFORMWINDOWCANVAS
#define INCL_ODAPI
#include <os2.h>
```

```
ODFacet      *facet;
HPS          rv;
```

```
rv = GetPS(facet);
```

-----

## GetPS (OS/2) Parameter - facet

**facet** (ODFacet \*) - input

The facet for which the PS is being obtained. This parameter is ignored by the [ODPlatformCanvas](#) class. For the derived class [ODPlatformWindowCanvas](#), it is used to obtain a PS for the specified facet's facet window.

-----

## GetPS (OS/2) Return Value - rv

**rv** ([HPS](#)) - returns  
A handle to a GPI PS specified in the call to the facet's [CreatePlatformCanvas](#) method.

---

## GetPS (OS/2) - Parameters

**facet** (ODFacet \*) - input  
The facet for which the PS is being obtained. This parameter is ignored by the [ODPlatformCanvas](#) class. For the derived class [ODPlatformWindowCanvas](#), it is used to obtain a PS for the specified facet's facet window.

**rv** ([HPS](#)) - returns  
A handle to a GPI PS specified in the call to the facet's [CreatePlatformCanvas](#) method.

---

## GetPS (OS/2) - Remarks

This method returns a GPI presentation space (PS) for the specified facet. You call this method in your part's [Draw](#) method when you need to draw your part's content. If this object's [SetPS](#) method has not been called to set a non-cached PS for the specified facet, this method returns a cached PS, obtained by calling the WinGetPS method for the facet window of the specified facet, which must be released by calling the platform canvas's [ReleasePS](#) method. Do not call WinReleasePS yourself for the cached PS. The GetPS method increments a reference count for the number of times it is called for each facet, and the [ReleasePS](#) method decrements this reference count. The cached PS is released (by calling the WinReleasePS method) when the reference count is decremented to zero.

See the "Facet Windows and Using Embedded Controls" recipe in the *OpenDoc Programming Guide* for more information about facet windows.

---

## GetPS (OS/2) - Topics

**Class:**  
[ODPlatformWindowCanvas](#)

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

## GetWindow (OS/2)

---

## GetWindow (OS/2) - Syntax

This method returns a reference to the window that was specified when this facet was created (by calling the facet's the [CreatePlatformWindowCanvas](#) method).

```
#define INCL_ODPLATFORMWINDOWCANVAS
#define INCL_ODAPI
#include <os2.h>

ODWindow      *rv;

rv = GetWindow();
```

---

## GetWindow (OS/2) Return Value - rv

**rv** (ODWindow \*) - returns  
A reference to the window object.

---

## GetWindow (OS/2) - Parameters

**rv** (ODWindow \*) - returns  
A reference to the window object.

---

## GetWindow (OS/2) - Topics

**Class:**  
ODPlatformWindowCanvas

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## HasWindow (OS/2)

---

## HasWindow (OS/2) - Syntax

This method indicates whether a given [ODPlatformCanvas](#) reference refers to an instance of the [ODPlatformCanvas](#) or [ODPlatformWindowCanvas](#) class.

```
#define INCL_ODPLATFORMWINDOWCANVAS
#define INCL_ODAPI
#include <os2.h>
```

```
ODBoolean    rv;  
  
rv = HasWindow();
```

---

## HasWindow (OS/2) Return Value - rv

**rv** ([ODBoolean](#)) - returns  
This method returns the following value:

kODTrue      The object is an instance of the [ODPlatformCanvas](#) class.

---

## HasWindow (OS/2) - Parameters

**rv** ([ODBoolean](#)) - returns  
This method returns the following value:

kODTrue      The object is an instance of the [ODPlatformCanvas](#) class.

---

## HasWindow (OS/2) - Topics

**Class:**  
ODPlatformWindowCanvas

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)

---

## ReleasePS (OS/2)

---

## ReleasePS (OS/2) - Syntax

This method decrements the reference count and releases the PS when the reference count is decremented to zero if the corresponding [GetPS](#) method for the specified facet returned a cached, micro PS.

```
#define INCL_ODPLATFORMWINDOWCANVAS  
#define INCL_ODAPI  
#include <os2.h>
```

```
ODFacet      *facet;  
  
ReleasePS(facet);
```

---

## ReleasePS (OS/2) Parameter - facet

**facet** (ODFacet \*) - input  
The facet passed to the corresponding [GetPS](#) method.

---

## ReleasePS (OS/2) - Return Value

None.

---

## ReleasePS (OS/2) - Parameters

**facet** (ODFacet \*) - input  
The facet passed to the corresponding [GetPS](#) method.

None.

---

## ReleasePS (OS/2) - Remarks

If this object's [SetPS](#) method has not been called to set a non-cached PS for the facet, then this method decrements the reference count of the number of times the [GetPS](#) method was called. When the reference count is decremented to zero, the cached PS for the facet window of this facet is released (by calling WinReleasePS). If the [SetPS](#) method has been called to set a non-cached PS for this facet, this method performs no action.

---

## ReleasePS (OS/2) - Topics

**Class:**  
ODPlatformWindowCanvas

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

# SetPS (OS/2)

---

## SetPS (OS/2) - Syntax

This method sets a non-cached presentation space (PS) on a facet window.

```
#define INCL_ODPLATFORMWINDOWCANVAS
#define INCL_ODAPI
#include <os2.h>

HPS          hps;
ODFacet      *facet;

SetPS(hps, facet);
```

---

## SetPS (OS/2) Parameter - hps

**hps** (**HPS**) - input

A handle to the PS with a window device context (DC). The window DC must be for the facet window of the specified window.

---

## SetPS (OS/2) Parameter - facet

**facet** (ODFacet \*) - input

A facet whose facet window is to be associated with the specified PS.

---

## SetPS (OS/2) - Return Value

None.

---

## SetPS (OS/2) - Parameters

**hps** (**HPS**) - input

A handle to the PS with a window device context (DC). The window DC must be for the facet window of the specified window.



**facet** (ODFacet \*) - input

A facet whose facet window is to be associated with the specified PS.

None.

-----

## SetPS (OS/2) - Remarks

This method sets a non-cached mirco or normal PS to be used for the facet window of the specified facet. It is the responsibility of the caller to create the PS and associate it with the window DC of the facet window. After this call is made with a non-cached PS handle, the [GetPS](#) method always returns the specified handle. If zero is specified for the PS handle, any previously specified non-cached PS is removed, and the [GetPS](#) method again returns a handle to a cached PS. It is the caller's responsibility to dispose of any non-cached PS that is removed in this way.

It is an error to call this method if there is an outstanding cached PS that has not been released. For this reason, it is best to call this method from your part's [FacetAdded for=textonly](#) method.

See the "Facet Windows and Using Embedded Controls" recipe for more information about setting a non-cached PS for a facet window.

-----

## SetPS (OS/2) - Exception Handling

kODErrInvalidParameter

The *hps* parameter has the wrong units or is associated with the wrong DC.

-----

## SetPS (OS/2) - Topics

### Class:

ODPlatformWindowCanvas

Select an item:

[Syntax](#)

[Parameters](#)

[Returns](#)

[Remarks](#)

[Exception Handling](#)

-----

## ODPopup

**Class Definition File:** POPUP.IDL

### Class Hierarchy

SOMObject

ODObject

ODRefCntObject

ODBaseMenuBar

ODMenuBar

**ODPopup**

### Description

This class enables parts to have a base pop-up menu. Parts obtain an instance to this class using the window state's [CopyBaseMenuBar](#) method. The ODPopup instance received back contains a base pop-up menu with the items **Open as**, **Settings**, **Show as**, and **Help**. Parts can then add or modify this menu.

## Methods

The methods defined by the ODPopup class include:

- [AddDefaultMenuItemBefore](#)
- [AddDefaultMenuItemLast](#)
- [CopyPopup](#)

## Overridden Methods

There are currently no methods overridden by the ODPopup class.

---

# AddDefaultMenuItemBefore (OS/2)

---

## AddDefaultMenuItemBefore (OS/2) - Syntax

This method adds one of the predefined OpenDoc menu items to the pop-up menu.

```
#define INCL_ODPOPUP
#define INCL_ODAPI
#include <os2.h>

ODMenuItemID    menuItemID;
ODMenuItemID    beforeID;
ODBoolean       rv;

rv = AddDefaultMenuItemBefore(menuItemID,
                              beforeID);
```

---

## AddDefaultMenuItemBefore (OS/2) Parameter - menuItemID

**menuItemID** ([ODMenuItemID](#)) - input  
A predefined OpenDoc menu ID, such as EDIT\_CUT.

---

## AddDefaultMenuItemBefore (OS/2) Parameter - beforeID

**beforeID** ([ODMenuItemID](#)) - input  
The menu item before which the new item is to be inserted.

---

## AddDefaultMenuItemBefore (OS/2) Return Value - rv

**rv** ([ODBoolean](#)) - returns  
Success indicator.

kODTrue	Successful completion
kODFalse	Error occurred

-----

## AddDefaultMenuItemBefore (OS/2) - Parameters

**menuItemID** ([ODMenuItemID](#)) - input  
A predefined OpenDoc menu ID, such as EDIT\_CUT.

**beforeID** ([ODMenuItemID](#)) - input  
The menu item before which the new item is to be inserted.

**rv** ([ODBoolean](#)) - returns  
Success indicator.

kODTrue	Successful completion
kODFalse	Error occurred

-----

## AddDefaultMenuItemBefore (OS/2) - Topics

**Class:**  
ODPopup

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)

-----

## AddDefaultMenuItemLast (OS/2)

-----

## AddDefaultMenuItemLast (OS/2) - Syntax

This method adds one of the predefined OpenDoc menu items to the end of the pop-up menu.

```
#define INCL_ODPOPUP
#define INCL_ODAPI
#include <os2.h>
```

```
ODMenuItemID    menuItemID;
ODBoolean        rv;
```

```
rv = AddDefaultMenuItemLast (menuItemID);
```

-----

## AddDefaultMenuItemLast (OS/2) Parameter - menuItemID

**menuItemID** ([ODMenuItemID](#)) - input  
A predefined OpenDoc menu ID, such as EDIT\_CUT.

-----

## AddDefaultMenuItemLast (OS/2) Return Value - rv

**rv** ([ODBoolean](#)) - returns  
Success indicator.

kODTrue	Successful completion
kODFalse	Error occurred

-----

## AddDefaultMenuItemLast (OS/2) - Parameters

**menuItemID** ([ODMenuItemID](#)) - input  
A predefined OpenDoc menu ID, such as EDIT\_CUT.

**rv** ([ODBoolean](#)) - returns  
Success indicator.

kODTrue	Successful completion
kODFalse	Error occurred

-----

## AddDefaultMenuItemLast (OS/2) - Topics

**Class:**  
ODPopup

Select an item:

[Syntax](#)  
[Parameters](#)  
[Returns](#)

-----

## CopyPopup (OS/2)

---

## CopyPopup (OS/2) - Syntax

This method clones this pop-up menu object.

```
#define INCL_ODPOPUP
#define INCL_ODAPI
#include <os2.h>

ODPopup      *rv;

rv = CopyPopup();
```

---

## CopyPopup (OS/2) Return Value - rv

**rv** (ODPopup \*) - returns

---

## CopyPopup (OS/2) - Parameters

**rv** (ODPopup \*) - returns

---

## CopyPopup (OS/2) - Remarks

Under normal circumstances, parts do not call this method. Instead, the window state's [CopyBasePopup](#) method is usually called.

---

## CopyPopup (OS/2) - Topics

**Class:**  
ODPopup

Select an item:  
[Syntax](#)  
[Parameters](#)  
[Returns](#)  
[Remarks](#)

---

